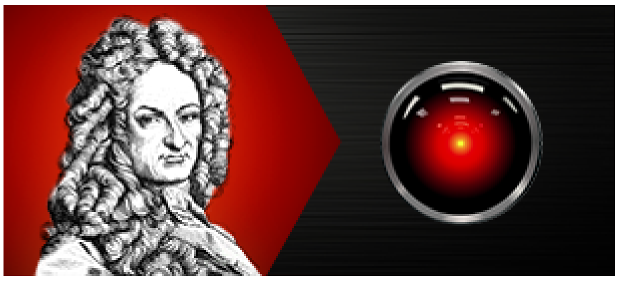
This essay is also in:

Data-Driven Law: Data Analytics and the New Legal Services » *WIRED »*

Computational Law, Symbolic Discourse and the AI Constitution

October 12, 2016

Leibniz's Dream



Gottfried Leibniz—who died 300 years ago this November—worked on many things. But a theme that recurred throughout his life was the goal of turning human law into an exercise in computation. Of course, as we know, he didn't succeed. But three centuries later, I think we're finally ready to give it a serious try again. And I think it's a really important thing to do—not just because it'll enable all sorts of new societal opportunities and structures, but because I think it's likely to be critical to the future of our civilization in its interaction with artificial intelligence.

Human law, almost by definition, dates from the very beginning of civilization—and undoubtedly it's the first system of rules that humans ever systematically defined. Presumably it was a model for the axiomatic structure of mathematics as defined by the likes of Euclid. And when science came along, "natural laws" (as their name suggests) were at first viewed as conceptually similar to human laws, except that they were supposed to define constraints for the universe (or God) rather than for humans.

Over the past few centuries we've had amazing success formalizing mathematics and exact science. And out of this there's a more general idea that's emerged: the idea of computation. In computation, we're dealing with arbitrary systems of rules—not necessarily ones that correspond to mathematical concepts we know, or features of the world we've identified. So now the question is: can we use the ideas of computation, in very much the way Leibniz imagined, to formalize human law?

The basic issue is that human law talks about human activities, and (unlike say for the mechanics of particles) we don't have a general formalism for describing human activities. When it comes to talking about money, for example, we often can be precise. And as a result, it's pretty easy to write a very formal contract for paying a subscription, or determining how an option on a publicly traded stock should work.

But what about all the things that typical legal contracts deal with? Well, clearly we have one way to write legal contracts: just use natural language (like English). It's often very stylized natural language, because it's trying to be as precise as possible. But ultimately it's never going to be precise. Because at the lowest level it's always going to depend on the meanings of words, which for natural language are effectively defined just by the practice and experience of the users of the language.

A New Kind of Language

For a computer language, though, it's a different story. Because now the constructs in the language are absolutely precise: instead of having a vague, societally defined effect on human brains, they're defined to have a very specific effect on a computer. Of course, traditional computer languages don't directly talk about things relevant to human activities: they only directly talk about things like setting values for variables, or calling abstractly defined functions.

But what I'm excited about is that we're starting to have a bridge between the precision of traditional computer languages and the ability to talk about real-world constructs. And actually, it's something I've personally been working on for more than three decades now: our knowledge-based Wolfram Language.

The Wolfram Language is precise: everything in it is defined to the point where a computer can unambiguously work with it. But its unique feature among computer languages is that it's knowledge based. It's not just a language to describe the low-level operations of a computer; instead, built right into the language is as much knowledge as possible about the real world. And this means that the language includes not just numbers like 2.7 and strings like "abc", but also constructs like the United States, or the Consumer Price Index, or an elephant. And that's exactly what we need in order to start talking about the kinds of things that appear in legal contracts or human laws.

I should make it clear that the Wolfram Language as it exists today doesn't include everything that's needed. We've got a large and solid framework, and we're off to a good start. But there's more about the world that we have to encode to be able to capture the full range of human activities and human legal specifications.

The Wolfram Language has, for example, a definition of what a banana is, broken down by all kinds of details. So if one says "you should eat a banana", the language has a way to represent "a banana". But as of now, it doesn't have a meaningful way to represent "you", "should" or "eat".

Is it possible to represent things like this in a precise computer language? Absolutely! But it takes language design to set up how to do it. Language design is a difficult business—in fact, it's probably the most intellectually demanding thing I know, requiring a strange mixture of high abstraction together with deep knowledge and down-to-earth practical judgment. But I've been doing it now for nearly four decades, and I think I'm finally ready for the challenge of doing language design for everyday discourse.

So what's involved? Well, let's first talk about it in a simpler case: the case of mathematics. Consider the function **Plus**, which adds things like numbers together. When we use the English word "plus" it can have all sorts of meanings. One of those meanings is adding numbers together. But there are other meanings, that are related, say, by various analogies ("product X plus", "the plus wire", "it's a real plus", ...).

When we come to define **Plus** in the Wolfram Language we want to build on the everyday notion of "plus", but we want to make it precise. And we can do that by picking the specific meaning of "plus" that's about adding things like numbers together. And once we know that this is what **Plus** means, we immediately know all sorts of properties, and can do explicit computations with it.

Now consider a concept like "magnesium". It's not as perfect and abstract a concept as **Plus**. But physics and chemistry give us a clear definition of the element magnesium—which we can then use in the Wolfram Language to have a well-defined "magnesium" entity.

It's very important that the Wolfram Language is a symbolic language—because it means that the things in it don't immediately have to have "values"; they can just be symbolic constructs that stand for themselves. And so, for example, the entity "magnesium" is represented as a symbolic construct, that doesn't itself "do" anything, but can still appear in a computation, just like, for example, a number (like 9.45) can appear.

There are many kinds of constructs that the Wolfram Language supports. Like "New York City" or "last Christmas" or "geographically contained within". And the point is that the design of the language has defined a precise meaning for them. New York City, for example, is taken to mean the precise legal entity considered to be New York City, with geographical borders defined by law. Internal to the Wolfram Language, there's always a precise canonical representation for something like New York City (it's `Entity["City", {"NewYork", "NewYork", "UnitedStates"}]`). And this internal representation is all that matters when it comes to computation. Yes, it's convenient to refer to New York City as "nyc", but in the Wolfram Language that natural language form is immediately converted to the precise internal form.

So what about "you should eat a banana"? Well, we've got to go through the same language design process for something like "eat" as for `Plus` (or "banana"). And the basic idea is that we've got to figure out a standard meaning for "eat". For example, it might be "ingestion of food by a person (or animal)". Now, there are plenty of other possible meanings for the English word "eat"—for example, ones that use analogies, as in "this function eats its arguments". But the idea—like for `Plus`—is to ignore these, and just to define a standard notion of "eat" that is precise, and suitable for computation.

One gets a reasonable idea of what kinds of constructs one has to deal with just by thinking about parts of speech in English. There are nouns. Sometimes (as in "banana" or "elephant") there's a pretty precise definition of what these correspond to, and usually the Wolfram Language already knows about them. Sometimes it's a little vaguer but still concrete (as in "chair" or "window"), and sometimes it's abstract (like "happiness" or "justice"). But in each case one can imagine one or several entities that capture a definite meaning for the noun—just like the Wolfram Language already has entities for thousands of kinds of things.

Beyond nouns, there are verbs. There's typically a certain superstructure that exists around verbs. Grammatically there might be a subject for the verb, and an object, and so on. Verbs are similar to functions in the Wolfram Language: each one deals with certain arguments, that for example correspond to its subject, object, etc. Now of course in English (or any other natural language) there are all sorts of elaborate special cases and extra features that can be associated with verbs. But basically we don't care about these. Because we're really just trying to define symbolic constructs that represent certain concepts. We don't have to capture every detail of how a particular verb works; we're just using the English verb as a way to give us a kind of "cognitive hook" for the concept.

We can go through other parts of speech. Adverbs that modify verbs; adjectives that modify nouns. These can sometimes be represented in the Wolfram Language by constructs like `EntityInstance`, and sometimes by options to functions. But the important point in all cases is that we're not trying to faithfully reproduce how the natural language works; we're just using the natural language as a guide to how concepts are set up.

Pronouns are interesting. They work a bit like variables in pure anonymous functions. In "you should eat a banana", the "you" is like a free variable that's going to be filled in with a particular person.

Parts of speech and grammatical structures suggest certain general features to capture in a symbolic representation of discourse. There are a bunch of others, though. For example, there are what amount to "calculi" that one needs to represent notions of time ("within the time interval", "starting later", etc.) or of space ("on top of", "contained within", etc.). We've already got many calculi like these in the Wolfram Language; the most straightforward are ones about numbers ("greater than", etc.) or sets ("member of"), etc. Some calculi have long histories ("temporal logic", "set theory", etc.); others still have to be constructed.

Is there a global theory of what to do? Well, no more than there's a global theory of how the world works. There are concepts and constructs that are part of how our world works, and we need to capture these. No doubt there'll be new things that come along in the future, and we'll want to capture those too. And my experience from building Wolfram|Alpha is that the best thing to do is just to build each thing one needs, without starting off with any kind of global theory. After a while, one may notice that one's built similar things several times, and one may go in and unify them.

One can get deep into the foundations of science and philosophy about this. Yes, there's a computational universe out there of all the possible rules by which systems can operate (and, yes, I've spent a good part of my life studying the basic science of this). And there's our physical universe that presumably operates according to certain rules from the computational universe. But from these rules can emerge all sorts of complex behavior, and in fact the phenomenon of computational irreducibility implies that in a sense there's no limit to what can be built up.

But there's not going to be an overall way to talk about all this stuff. And if we're going to be dealing with any finite kind of discourse it's going to only capture certain features. Which features we choose to capture is going to be determined by what concepts have evolved in the history of our society. And usually these concepts will be mirrored in the words that exist in the languages we use.

At a foundational level, computational irreducibility implies that there'll always be new concepts that could be introduced. Back in antiquity, Aristotle introduced logic as a way to capture certain aspects of human discourse. And there are other frameworks that have been introduced in the history of philosophy, and more recently, natural language processing and artificial intelligence research. But computational irreducibility effectively implies that none of them can ever ultimately be complete. And we must expect that as the concepts we consider relevant evolve, so too must the symbolic representation we have for discourse.

The Discourse Workflow

OK, so let's say we've got a symbolic representation for discourse. How's it actually going to be used? Well, there are some good clues from the way natural language works.

In standard discussions of natural language, it's common to talk about "interrogative statements" that ask a question, "declarative statements" that assert something and "imperative statements" that say to do something. (Let's ignore "exclamatory statements", like expletives, for now.)

Interrogative statements are what we're dealing with all the time in Wolfram|Alpha: "what is the density of gold?", "what is 3+7?", "what was the latest reading from that sensor?", etc. They're also common in notebooks used to interact with the Wolfram Language: there's an input (**In[1]:= 2+2**) and then there's a corresponding output (**Out[1]= 4**).

Declarative statements are all about filling in particular values for variables. In a very coarse way, one can set values (**x=7**), as in typical procedural languages. But it's typically better to think about having environments in which one's asserting things. Maybe those environments are supposed to represent the real world, or some corner of it. Or maybe they're supposed to represent some fictional world, where for example dinosaurs didn't go extinct, or something.

Imperative statements are about making things happen in the world: "open the pod bay doors", "pay Bob 0.23 bitcoin", etc.

In a sense, interrogative statements determine the state of the world, declarative statements assert things about the state of the world, and imperative statements change the state of the world.

In different situations, we can mean different things by "the world". We could be talking about abstract constructs, like integers or logic operations, that just are the way they are. We could be talking about natural laws or other features of our physical universe that we can't change. Or we could be talking about our local environment, where we can move around tables and chairs, choose to eat bananas, and so on. Or we could be talking about our mental states, or the internal state of something like a computer.

There are lots of things one can do if one has a general symbolic representation for discourse. But one of them—which is the subject of this post—is to express things like legal contracts. The beginning of a contract, with its various whereas clauses, recitals, definitions and so on tends to be dense with declarative statements ("this is so"). Then the actual terms of the contract tend to end up with imperative statements ("this should happen"), perhaps depending on certain things determined by interrogative statements ("did this happen?").

It's not hard to start seeing the structure of contracts as being much like programs. In simple cases, they just contain logical conditionals: "if X then Y". In other cases they're more modeled on math: "if this amount of X happens, that amount of Y should happen". Sometimes there's iteration: "keep doing X until Y happens". Occasionally there's some recursion: "keep applying X to every Y". And so on.

There are already some places where legal contracts are routinely represented by what amount to programs. The most obvious are financial contracts for things like bonds and options—which just amount to little programs that define payouts based on various formulas and conditionals.

There's a whole industry of using "rules engines" to encode certain kinds of regulations as "if then" rules, usually mixed with formulas. In fact, such things are almost universally used for tax and insurance computations. (They're also common in pricing engines and the like.)

Of course, it's no coincidence that one talks about "legal codes". The word code—which comes from the Latin *codex*—originally referred to systematic collections of legal rules. And when programming came along a couple of millennia later, it used the word "code" because it basically saw itself as similarly setting up rules for how things should work, except now the things had to do with the operation of computers rather than the conduct of worldly affairs.

But now, with our knowledge-based computer language and the idea of a symbolic discourse language, what we're trying to do is to make it so we can talk about a broad range of worldly affairs in the same kind of way that we talk about computational processes—so we put all those legal codes and contracts into computational form.

Code versus Language

How should we think about symbolic discourse language compared to ordinary natural language? In a sense, the symbolic discourse language is a representation in which all the nuance and “poetry” have been “crushed” out of the natural language. The symbolic discourse language is precise, but it’ll almost inevitably lose the nuance and poetry of the original natural language.

If someone says “2+2” to Wolfram|Alpha, it’ll dutifully answer “4”. But what if instead they say, “hey, will you work out 2+2 for me”. Well, that sets up a different mood. But Wolfram|Alpha will take that input and convert it to exactly the same symbolic form as “2+2”, and similarly just respond “4”.

This is exactly the kind of thing that’ll happen all the time with symbolic discourse language. And if the goal is to answer precise questions—or, for that matter, to create a precise legal contract, it’s exactly what one wants. One just needs the hard content that will actually have a consequence for what one’s trying to do, and in this case one doesn’t need the “extras” or “pleasantries”.

Of course, what one chooses to capture depends on what one’s trying to do. If one’s trying to get psychological information, then the “mood” of a piece of natural language can be very important. Those “exclamatory statements” (like expletives) carry meaning one cares about. But one can still perfectly well imagine capturing things like that in a symbolic way—for example by having an “emotion track” in one’s symbolic discourse language. (Very coarsely, this might be represented by sentiment or by position in an emotion space—or, for that matter, by a whole symbolic language derived, say, from emoji.)

In actual human communication through natural language, “meaning” is a slippery concept, that inevitably depends on the context of the communication, the history of whoever is communicating, and so on. My notion of a symbolic discourse language isn’t to try to magically capture the “true meaning” of a piece of natural language. Instead, my goal is just to capture some meaning that one can then compute with.

For convenience, one might choose to start with natural language, and then try to translate it into the symbolic discourse language. But the point is for the symbolic discourse language to be the real representation: the natural language is just a guide for trying to generate it. And in the end, the notion is that if one really wants to be sure one’s accurate in what one’s saying, one should say it directly in the symbolic discourse language, without ever using natural language.

Back in the 1600s, one of Leibniz's big concerns was to have a representation that was independent of which natural language people were using (French, German, Latin, etc.). And one feature of a symbolic discourse language is that it has to operate "below" the level of specific natural languages.

There's a rough kind of universality among human languages, in that it seems to be possible to represent any human concept at least to some approximation in any language. But there are plenty of nuances that are extremely hard to translate—between different languages, or the different cultures that surround them (or even the same language at different times in history). But in the symbolic discourse language, one's effectively "crushing out" these differences—and getting something that is precise, even though it typically won't correspond exactly to any particular human natural language.

A symbolic discourse language is about representing things in the world. Natural language is just one way to try to describe those things. But there are others. For example, one might give a picture. One could try to describe certain features of the picture in natural language ("a cat with a hat on its head")—or one could go straight from the picture to the symbolic discourse language.

In the example of a picture, it's very obvious that the symbolic discourse language isn't going to capture everything. Maybe it could capture something like "he is taking the diamond". But it's not going to specify the color of every pixel, and it's not going to describe all conceivable features of a scene at every level of detail.

In some sense, what the symbolic discourse language is doing is to specify a model of the system it's describing. And like any model, it's capturing some features, and idealizing others away. But the importance of it is that it provides a solid foundation on which computations can be done, conclusions can be drawn, and actions can be taken.

Why Now?

I've been thinking about creating what amounts to a general symbolic discourse language for nearly 40 years. But it's only recently—with the current state of the Wolfram Language—that I've had the framework to actually do it. And it's also only recently that I've understood how to think about the problem in a sufficiently practical way.

Yes, it's nice in principle to have a symbolic way to represent things in the world. And in specific cases—like answering questions in Wolfram|Alpha—it's completely clear why it's worth doing this. But what's the point of dealing with more general discourse? Like, for example, when do we really want to have a "general conversation" with a machine?

The Turing test says that being able to do this is a sign of achieving general AI. But "general conversations" with machines—without any particular purpose in mind—so far usually seem in practice to devolve quickly into party tricks and Easter eggs. At least that's our experience looking at interactions people have with Wolfram|Alpha, and it also seems to be the experience with decades of chatbots and the like.

But the picture quickly changes if there's a purpose to the conversation: if you're actually trying to get the machine to do something, or learn something from the machine. Still, in most of these cases, there's no real reason to have a general representation of things in the world; it's sufficient just to represent specific machine actions, particular customer service goals, or whatever. But if one wants to tackle the general problem of law and contracts, it's a different story. Because inevitably one's going to have to represent the full spectrum of human affairs and issues. And so now there's a definite goal to having a symbolic representation of the world: one needs it to be able to say what should happen and have machines understand it.

Sometimes it's useful to do that because one wants the machines just to be able to check whether what was supposed to happen actually did; sometimes one wants to actually have the machines automatically enforce or do things. But either way, one needs the machine to be able to represent general things in the world—and so one needs a symbolic discourse language to be able to do this.

Some History

In a sense, it's a very obvious idea to have something like a symbolic discourse language. And indeed it's an idea that's come up repeatedly across the course of centuries. But it's proved a very difficult idea to make work, and it has a history littered with (sometimes quite wacky) failures.

Things in a sense started well. Back in antiquity, logic as discussed by Aristotle provided a very restricted example of a symbolic discourse language. And when the formalism of mathematics began to emerge it provided another example of a restricted symbolic discourse language.

But what about more general concepts in the world? There'd been many efforts—between the Tetractys of the Pythagoreans and the I Ching of the Chinese—to assign symbols or numbers to a few important concepts. But around 1300 Ramon Llull took it further, coming up with a whole combinatorial scheme for representing concepts—and then trying to implement this with circles of paper that could supposedly mechanically determine the validity of arguments, particularly religious ones.

Four centuries later, Gottfried Leibniz was an enthusiast of Llull's work, at first imagining that perhaps all concepts could be converted to numbers and truth then determined by doing something like factoring into primes. Later, Leibniz starting talking about a *characteristica universalis* (or, as Descartes called it, an "alphabet of human thoughts")—essentially a universal symbolic language. But he never really tried to construct such a thing, instead chasing what one might consider "special cases"—including the one that led him to calculus.

With the decline of Latin as the universal natural language in the 1600s, particularly in areas like science and diplomacy, there had already been efforts to invent "philosophical languages" (as they were called) that would represent concepts in an abstract way, not tied to any specific natural language. The most advanced of these was by John Wilkins—who in 1668 produced a book cataloging over 10,000 concepts and representing them using strange-looking glyphs, with a rendering of the Lord's Prayer as an example.

In some ways these efforts evolved into the development of encyclopedias and later thesauruses, but as language-like systems, they basically went nowhere. Two centuries later, though, as the concept of internationalization spread, there was a burst of interest in constructing new, country-independent languages—and out of this emerged Volapük and then Esperanto. These languages were really just artificial natural languages; they weren't an attempt to produce anything like a symbolic discourse language. I always used to enjoy seeing signs in Esperanto at European airports, and was disappointed in the 1980s when these finally disappeared. But, as it happens, right around that time, there was another wave of language construction. There were languages like Lojban, intended to be as unambiguous as possible, and ones like the interestingly minimal Toki Pona intended to support the simple life, as well as the truly bizarre Ithkuil, intended to encompass the broadest range of linguistic and supposedly cognitive structures.

Along the way, there were also attempts to simplify languages like English by expressing everything in terms of 1000 or 2000 basic words (instead of the usual 20,000–30,000) —as in the "Simple English" version of Wikipedia or the xkcd Thing Explainer.

There were a few, more formal, efforts. One example was Hans Freudenthal's 1960 Lincos "language for cosmic intercourse" (i.e. communication with extraterrestrials) which attempted to use the notation of mathematical logic to capture everyday concepts. In the early days of the field of artificial intelligence, there were plenty of discussions of "knowledge representation", with approaches based variously on the grammar of natural language, the structure of predicate logic or the formalism of databases. Very few large-scale projects were attempted (Doug Lenat's Cyc being a notable counterexample), and when I came to develop Wolfram|Alpha I was disappointed at how little of relevance to our needs seemed to have emerged.

In a way I find it remarkable that something as fundamental as the construction of a symbolic discourse language should have had so little serious attention paid to it in the past. But at some level it's not so surprising. It's a difficult, large project, and it somehow lies in between established fields. It's not a linguistics project. Yes, it may ultimately illuminate how languages work, but that's not its main point. It's not a computer science project because it's really about content, not algorithms. And it's not a philosophy project because it's mostly about specific nitty-gritty and not much about general principles.

There've been a few academic efforts in the last half century or so, discussing ideas like "semantic primes" and "natural semantic metalanguage". Usually such efforts have tried to attach themselves to the field of linguistics—but their emphasis on abstract meaning rather than pure linguistic structure has put them at odds with prevailing trends, and none have turned into large-scale projects.

Outside of academia, there's been a steady stream of proposals—sometimes promoted by wonderfully eccentric individuals—for systems to organize and name concepts in the world. It's not clear how far this pursuit has come since Ramon Llull—and usually it's only dealing with pure ontology, and never with full meaning of the kind that can be conveyed in natural language.

I suppose one might hope that with all the recent advances in machine learning there'd be some magic way to automatically learn an abstract representation for meaning. And, yes, one can take Wikipedia, for example, or a text corpus, and use dimension reduction to derive some effective "space of concepts". But, not too surprisingly, simple Euclidean space doesn't seem to be a very good model for the way concepts relate (one can't even faithfully represent graph distances). And even the problem of taking possible meanings for words—as a dictionary might list them—and breaking them into clusters in a space of concepts doesn't seem to be easy to do effectively.

Still, as I'll discuss later, I think there's a very interesting interplay between symbolic discourse language and machine learning. But for now my conclusion is that there's not going to be any alternative but to use human judgment to construct the core of any symbolic discourse language that's intended for humans to use.

Contracts into Code

But let's get back to contracts. Today, there are hundreds of billions of them being signed every year around the world (and vastly more being implicitly entered into)—though the number of "original" ones that aren't just simple modifications is probably just in the millions (and is perhaps comparable to the number of original computer programs or apps being written.)

So can these contracts be represented in precise symbolic form, as Leibniz hoped 300 years ago? Well, if we can develop a decently complete symbolic discourse language, it should be possible. (Yes, every contract would have to be defined relative to some underlying set of "governing law" rules, etc., that are in some ways like the built-in functions of the symbolic discourse language.)

But what would it mean? Among other things, it would mean that contracts themselves would become computable things. A contract would be converted to a program in the symbolic discourse language. And one could do abstract operations just on this program. And this means one can imagine formally determining—in effect through a kind of generalization of logic—whether, say, a given contract has some particular implication, could ever lead to some particular outcome, or is equivalent to some other contract.

Ultimately, though, there's a theoretical problem with this. Because questions like this can run into issues of formal undecidability, which means there's no guarantee that any systematic finite computation will answer them. The same problem arises in reasoning about typical software programs people write, and in practice it's a mixed bag, with some things being decidable, and others not.

Of course, even in the Wolfram Language as it is today, there are plenty of things (such as the very basic "are these expressions equal?") that are ultimately in principle undecidable. And there are certainly questions one can ask that run right into such issues. But an awful lot of the kinds of questions that people naturally ask turn out to be answerable with modest amounts of computation. And I wouldn't be surprised if this were true for questions about contracts too. (It's worth noting that human-formulated questions tend to run into undecidability much less than questions picked, say at random, from the whole computational universe of possibilities.)

If one has contracts in computational form, there are other things one can expect to do too. Like to be able to automatically work out what the contracts imply for a large range of possible inputs. The 1980s revolution in quantitative finance started when it became clear one could automatically compute distributions of outcomes for simple options contracts. If one had lots (perhaps billions) of contracts in computational form, there'd be a lot more that could be done along these lines—and no doubt, for better or worse, whole new areas of financial engineering that could be developed.

Where Do the Inputs Come From?

OK, so let's say one has a computational contract. What can one directly do with it? Well, it depends somewhat on what the form of its inputs is. One important possibility is that they're in a sense "born computational": that they're immediately statements about a computational system ("how many accesses has this ID made today?", "what is the ping time for this connection?", "how much bitcoin got transferred?", etc.). And in that case, it should be possible to immediately and unambiguously "evaluate" the contract—and find out if it's being satisfied.

This is something that's very useful for lots of purposes—both for humans interacting with machines, and machines interacting with machines. In fact, there are plenty of cases where versions of it are already in use. One can think of computer security provisions such as firewall rules as one example. There are others that are gradually emerging, such as automated SLAs (service-level agreements) and automated terms-of-service. (I'm certainly hoping our company, for example, will be able to make these a routine part of our business practices before too long.)

But, OK, it's certainly not true that every input for every contract is "born computational": plenty of inputs have to come from seeing what happens in the "outside" world ("did the person actually go to place X?", "was the package maintained in a certain environment?", "did the information get leaked to social media?", "is the parrot dead?", etc.). And the first thing to say is that in modern times it's become vastly easier to automatically determine things about the world, not least because one can just make measurements with sensors. Check the GPS trace. Look at the car counting sensor. And so on. The whole Internet of Things is out there to provide input about the real world for computational contracts.

Having said this, though, there's still an issue. Yes, with a GPS trace there's a definite answer (assuming the GPS is working properly) for whether someone or something went to a particular place. But let's say one's trying to determine something less obviously numerical. Let's say, for example, that one's trying to determine whether a piece of fruit should be considered "Fancy Grade" or not. Well, given some pictures of the piece of fruit an expert can pretty unambiguously tell. But how can we make this computational?

Well, here's a place where we can use modern machine learning. We can set up some neural net, say in the Wolfram Language, and then show it lots of examples of fruit that's Fancy Grade and that's not. And from my experience (and those of our customers!) most of the time we'll get a system that's really good at a task like grading fruit. It'll certainly be much faster than humans, and it'll probably be more reliable and more consistent too.

And this gives a whole new way to set up contracts about things in the world. Two parties can just agree that the contract should say "if the machine learning system says X then do Y". In a sense it's like any other kind of computational contract: the machine learning system is just a piece of code. But it's a little different. Because normally one expects that one can readily examine everything that a contract says: one can in effect read and understand the code. But with machine learning in the middle, there can no longer be any expectation of that.

Nobody specifically set up all those millions of numerical weights in the neural net; they were just determined by some approximate and somewhat random process from whatever training data that was given. Yes, in principle we can measure everything about what's happening inside the neural net. But there's no reason to expect that we'll ever be able to get an understandable explanation—or prediction—of what the net will do in any particular case. Most likely it's an example of the phenomenon I call computational irreducibility–which means there really isn't any way to see what will happen much more efficiently than just by running it.

What's the difference with asking a human expert, then, whose thought processes one can't understand? Well, in practice machine learning is much faster so one can make much more use of "expert judgment". And one can set things up so they're repeatable, and one can for example systematically test for biases one thinks might be there, and so on.

Of course, one can always imagine cheating the machine learning. If it's repeatable, one could use machine learning itself to try to learn cases where it would fail. And in the end it becomes rather like computer security, where holes are being found, patches are being applied, and so on. And in some sense this is no different from the typical situation with contracts too: one tries to cover all situations, then it becomes clear that something hasn't been correctly addressed, and one tries to write a new contract to address it, and so on.

But the important bottom line is that with machine learning one can expect to get "judgment oriented" input into contracts. I expect the typical pattern will be this: in the contract there'll be something stated in the symbolic discourse language (like "X will personally do Y"). And at the level of the symbolic discourse language there'll be a clear meaning to this, from which, for example, all sorts of implications can be drawn. But then there's the question of whether what the contract said is actually what happened in the real world. And, sure, there can be lots of sensor data that gives information on this. But in the end there'll be a "judgment call" that has to be made. Did the person actually personally do this? Well—like for a remote exam proctoring system—one can have a camera watching the person, one can record their pattern of keystrokes, and maybe even measure their EEG. But something's got to synthesize this data, and make the judgment call about what happened, and turn this in effect into a symbolic statement. And in practice I expect it will typically end up being a machine learning system that does this.

Smart Contracts

OK, so let's say we've got ways to set up computational contracts. How can we enforce them? Well, ones that basically just involve computational processes can at some level enforce themselves. A particular piece of software can be built to issue licenses only in such-and-such a way. A cloud system can be built to make a download available only if it receives a certain amount of bitcoin. And so on.

But how far do we trust what's going on? Maybe someone hacked the software, or the cloud. How can we be sure nothing bad has happened? The basic answer is to use the fact that the world is a big place. As a (sometime) physicist it makes me think of measurement in quantum mechanics. If we're just dealing with a little quantum effect, there's always interference that can happen. But when we do a real measurement, we're amplifying that little quantum effect to the point where so many things (atoms, etc.) are involved that it's unambiguous what happened—in much the same way as the Second Law of Thermodynamics makes it inconceivable that all the air molecules in a room will spontaneously line up on one side.

And so it is with bitcoin, Ethereum, etc. The idea is that some particular thing that happened ("X paid Y such-and-such" or whatever) is shared and recorded in so many places that there can't be any doubt about it. Yes, it's in principle possible that all the few thousand places that actually participate in something like bitcoin today could collude to give a fake result. But the idea is that it's like with gas molecules in a room: the probability is inconceivably small. (As it happens, my Principle of Computational Equivalence suggests that there's more than an analogy with the gas molecules, and that actually the underlying principles at work are basically exactly the same. And, yes, there are lots of interesting technical details about the operation of distributed blockchain ledgers, distributed consensus protocols, etc., but I'm not going to get into them here.)

It's popular these days to talk about "smart contracts". When I've been talking about "computational contracts" I mean contracts that can be expressed computationally. But by "smart contracts" people usually mean contracts that can both be expressed computationally and execute automatically. Most often the idea is to set up a smart contract in a distributed computation environment like Ethereum, and then to have the code in the contract evaluate based on inputs from the computation environment.

Sometimes the input is intrinsic—like the passage of time (who could possibly tamper with the clock of the whole internet?), or physically generated random numbers. And in cases like this, one has fairly pure smart contracts, say for paying subscriptions, or for running distributed lotteries.

But more often there has to be some input from the outside—from something that happens in the world. Sometimes one just needs public information: the price of a stock, the temperature at a weather station, or a seismic event like a nuclear explosion. But somehow the smart contract needs access to an "oracle" that can give it this information. And conveniently enough, there is one good such oracle available in the world: Wolfram|Alpha. And indeed Wolfram|Alpha is becoming widely used as an oracle for smart contracts. (Yes, our general public terms of service say you currently just shouldn't rely on Wolfram|Alpha for anything you consider critical—though hopefully soon those terms of service will get more sophisticated, and computational.)

But what about non-public information from the outside world? The current thinking for smart contracts tends to be that one has to get humans in the loop to verify the information: that in effect one has to have a jury (or a democracy) to decide whether something is true. But is that really the best one can do? I tend to suspect there's another path, that's like using machine learning to inject human-like judgment into things. Yes, one can use people, with all their inscrutable and hard-to-systematically-influence behavior. But what if one replaces those people in effect by AIs—or even a collection of today's machine-learning systems?

One can think of a machine-learning system as being a bit like a cryptosystem. To attack it and spoof its input one has to do something like inverting how it works. Well, given a single machine-learning system there's a certain effort needed to achieve this. But if one has a whole collection of sufficiently independent systems, the effort goes up. It won't be good enough just to change a few parameters in the system. But if one just goes out into the computational universe and picks systems at random then I think one can expect to have the same kind of independence as by having different people. (To be fair, I don't yet quite know how to apply the mining of the computational universe that I've done for programs like cellular automata to the case of systems like neural nets.)

There's another point as well: if one has a sufficiently dense net of sensors in the world, then it becomes increasingly easy to be sure about what's happened. If there's just one motion sensor in a room, it might be easy to cover it. And maybe even if there are several sensors, it's still possible to avoid them, *Mission Impossible*-style. But if there are enough sensors, then by synthesizing information from them one can inevitably build up an understanding of what actually happened. In effect, one has a model of how the world works, and with enough sensors one can validate that the model is correct.

It's not surprising, but it always helps to have redundancy. More nodes to ensure the computation isn't tampered with. More machine-learning algorithms to make sure they aren't spoofed. More sensors to make sure they're not fooled. But in the end, there has to be something that says what should happen—what the contract is. And the contract has to be expressed in some language in which there are definite concepts. So somehow from the various redundant systems one has in the world, one has to make a definite conclusion—one has to turn the world into something symbolic, on which the contract can operate.

Writing Computational Contracts

Let's say we have a good symbolic discourse language. Then how should contracts actually get written in it?

One approach is to take existing contracts written in English or any other natural language, and try to translate (or parse) them into the symbolic discourse language. Well, what will happen is somewhat like what happens with Wolfram|Alpha today. The translator will not know exactly what the natural language was supposed to mean, and so it will give several possible alternatives. Maybe there was some meaning that the original writer of the natural-language contract had in mind. But maybe the "poetry" of that meaning can't be expressed in the symbolic discourse language: it requires something more definite. And a human is going to have to decide which alternative to pick.

Translating from natural-language contracts may be a good way to start, but I suspect it will quickly give way to writing contracts directly in the symbolic discourse language. Today lawyers have to learn to write legalese. In the future, they're going to have to learn to write what amounts to code: contracts expressed precisely in a symbolic discourse language.

One might think that writing everything as code, rather than natural-language legalese, would be a burden. But my guess is that it will actually be a great benefit. And it's not just because it will let contracts operate more easily. It's also that it will help lawyers think better about contracts. It's an old claim (the Sapir–Whorf hypothesis) that the language one uses affects the way one thinks. And this is no doubt somewhat true for natural languages. But in my experience it's dramatically true for computer languages. And indeed I've been amazed over the years at how my thinking has changed as we've added more to the Wolfram Language. When I didn't have a way to express something, it didn't enter my thinking. But once I had a way to express it, I could think in terms of it.

And so it will be, I believe, for legal thinking. When there's a precise symbolic discourse language, it'll become possible to think more clearly about all sorts of things.

Of course, in practice it'll help that there'll no doubt be all sorts of automated annotation: "if you add that clause, it'll imply X, Y and Z", etc. It'll also help that it'll routinely be possible to take some contract and simulate its consequences for a range of inputs. Sometimes one will want statistical results ("is this biased?"). Sometimes one will want to hunt for particular "bugs" that will only be found by trying lots of inputs.

Yes, one can read a contract in natural language, like one can read a math paper. But if one really wants to know its implications one needs it in computational form, so one can run it and see what it implies—and also so one can give it to a computer to implement.

The World with Computational Contracts

Back in ancient Babylon it was a pretty big deal when there started to be written laws like the Code of Hammurabi. Of course, with very few people able to read, there was all sorts of clunkiness at first—like having people recite the laws in order from memory. Over the centuries things got more streamlined, and then about 500 years ago, with the advent of widespread literacy, laws and contracts started to be able to get more complex (which among other things allowed them to be more nuanced, and to cover more situations).

In recent decades the trend has accelerated, particularly now that it's so easy to copy and edit documents of any length. But things are still limited by the fact that humans are in the loop, authoring and interpreting the documents. Back 50 years ago, pretty much the only way to define a procedure for anything was to write it down, and have humans implement it. But then along came computers, and programming. And very soon it started to be possible to define vastly more complex procedures—to be implemented not by humans, but instead by computers.

And so, I think, it will be with law. Once computational law becomes established, the complexity of what can be done will increase rapidly. Typically a contract defines some model of the world, and specifies what should happen in different situations. Today the logical and algorithmic structure of models defined by contracts still tends to be fairly simple. But with computational contracts it'll be feasible for them to be much more complex—so that they can for example more faithfully capture how the world works.

Of course, that just makes defining what should happen even more complex—and before long it might feel a bit like constructing an operating system for a computer, that tries to cover all the different situations the computer might find itself in.

In the end, though, one's going to have to say what one wants. One might be able to get a certain distance by just giving specific examples. But ultimately I think one's going to have to use a symbolic discourse language that can express a higher level of abstraction.

Sometimes one will be able to just write everything in the symbolic discourse language. But often, I suspect, one will use the symbolic discourse language to define what amount to goals, and then one will have to use machine-learning kinds of methods to fill in how to define a contract that actually achieves them.

And as soon as there's computational irreducibility involved, it'll typically be impossible to know for sure that there are no bugs, or "unintended consequences". Yes, one can do all kinds of automated tests. But in the end it's theoretically impossible to have any finite procedure that can guarantee to check all possibilities.

Today there are plenty of legal situations that are too complex to handle without expert lawyers. And in a world where computational law is common, it won't just be convenient to have computers involved, it'll be necessary.

In a sense it's similar to what's already happened in many areas of engineering. Back when humans had to design everything themselves, humans could typically understand the structures that were being built. But once computers are involved in design it becomes inevitable that they're needed in figuring out how things work too.

Today a fairly complex contract might involve a hundred pages of legalese. But once there's computational law—and particularly contracts constructed automatically from goals—the lengths are likely to increase rapidly. At some level it won't matter, though—just as it doesn't really matter how long the code of a program one's using is. Because the contract will in effect just be run automatically by computer.

Leibniz saw computation as a simplifying element in the practice of law. And, yes, some things will become simpler and better defined. But a vast ocean of complexity will also open up.

What Does It Mean for AIs?

How should one tell an AI what to do? Well, you have to have some form of communication that both humans and AIs can understand—and that is rich enough to describe what one wants. And as I've described elsewhere, what I think this basically means is that one has to have a knowledge-based computer language—which is precisely what the Wolfram Language is—and ultimately one needs a full symbolic discourse language.

But, OK, so one tells an AI to do something, like "go get some cookies from the store". But what one says inevitably won't be complete. The AI has to operate within some model of the world, and with some code of conduct. Maybe it can figure out how to steal the cookies, but it's not supposed to do that; presumably one wants it to follow the law, or a certain code of conduct.

And this is where computational law gets really important: because it gives us a way to provide that code of conduct in a way that AIs can readily make use of.

In principle, we could have AIs ingest the complete corpus of laws and historical cases and so on, and try to learn from these examples. But as AIs become more and more important in our society, it's going to be necessary to define all sorts of new laws, and many of these are likely to be "born computational", not least, I suspect, because they'll be too algorithmically complex to be usefully described in traditional natural language.

There's another problem too: we really don't just want AIs to follow the letter of the law (in whatever venue they happen to be), we want them to behave ethically too, whatever that may mean. Even if it's within the law, we probably don't want our AIs lying and cheating; we want them somehow to enhance our society along the lines of whatever ethical principles we follow.

Well, one might think, why not just teach AIs ethics like we could teach them laws? In practice, it's not so simple. Because whereas laws have been somewhat decently codified, the same can't be said for ethics. Yes, there are philosophical and religious texts that talk about ethics. But it's a lot vaguer and less extensive than what exists for law.

Still, if our symbolic discourse language is sufficiently complete, it certainly should be able to describe ethics too. And in effect we should be able to set up a system of computational laws that defines a whole code of conduct for AIs.

But what should it say? One might have a few immediate ideas. Perhaps one could combine all the ethical systems of the world. Obviously hopeless. Perhaps one could have the AIs just watch what humans do and learn their system of ethics from it. Similarly hopeless. Perhaps one could try something more local, where the AIs switch their behavior based on geography, cultural context, etc. (think "protocol droid"). Perhaps useful in practice, but hardly a complete solution.

So what can one do? Well, perhaps there are a few principles one might agree on. For example, at least the way we think about things today, most of us don't want humans to go extinct (of course, maybe in the future, having mortal beings will be thought too disruptive, or whatever). And actually, while most people think there are all sorts of things wrong with our current society and civilization, people usually don't want it to change too much, and they definitely don't want change forced upon them.

So what should we tell the AIs? It would be wonderful if we could just give the AIs some simple set of almost axiomatic principles that would make them always do what we want. Maybe they could be based on Asimov's Three Laws of Robotics. Maybe they could be something seemingly more modern based on some kind of global optimization. But I don't think it's going to be that easy.

The world is a complicated place; if nothing else, that's basically guaranteed by the phenomenon of computational irreducibility. And it's pretty much inevitable that there's not going to be any finite procedure that'll force everything to "come out the way one wants" (whatever that may be).

Let me take a somewhat abstruse, but well defined, example from mathematics. We think we know what integers are. But to really be able to answer all questions about integers (including about infinite collections of them, etc.) we need to set up axioms that define how integers work. And that's what Giuseppe Peano tried to do in the late 1800s. For a while it looked good, but then in 1931 Kurt Gödel surprised the world with his Incompleteness Theorem, which implied among other things, that actually, try as one might, there was never going to be a finite set of axioms that would define the integers as we expect them to be, and nothing else.

In some sense, Peano's original axioms actually got quite close to defining just the integers we want. But Gödel showed that they also allow bizarre non-standard integers, where for example the operation of addition isn't finitely computable.

Well, OK, that's abstract mathematics. What about the real world? Well, one of the things that we've learned since Gödel's time is that the real world can be thought of in computational terms, pretty much just like the mathematical systems Gödel considered. And in particular, one can expect the same phenomenon of computational irreducibility (which itself is closely related to Gödel's Theorem). And the result of this is that whatever simple intuitive goal we may define, it's pretty much inevitable we'll have to build up what amount to an arbitrarily complicated collection of rules to try to achieve it —and whatever we do, there'll always be at least some "unintended consequences".

None of this should really come as much of a surprise. After all, if we look at actual legal systems as they've evolved over the past couple of thousand years, there always end up being a lot of laws. It's not like there's a single principle from which everything else can be derived; there inevitably end up being lots of different situations that have to be covered.

Principles of the World?

But is all this complexity just a consequence of the "mechanics" of how the world works? Imagine—as one expects—that AIs get more and more powerful. And that more and more of the systems of the world, from money supplies to border controls, are in effect put in the hands of AIs. In a sense, then, the AIs play a role a little bit like governments, providing an infrastructure for human activities.

So, OK, perhaps we need a "constitution" for the AIs, just like we set up constitutions for governments. But again the question comes: what should the constitution have in it?

Let's say that the AIs could mold human society in pretty much any way. How would we want it molded? Well, that's an old question in political philosophy, debated since antiquity. At first an idea like utilitarianism might sound good: somehow maximize the well-being of as many people as possible. But imagine actually trying to do this with AIs that in effect control the world. Immediately one is thrust into concrete versions of questions that philosophers and others have debated for centuries. Let's say one can sculpt the probability distribution for happiness among people in the world. Well, now we've got to get precise about whether it's the mean or the median or the mode or a quantile or, for that matter, the kurtosis of the distribution that we're trying to maximize.

No doubt one can come up with rhetoric that argues for some particular choice. But there just isn't an abstract "right answer". Yes, we can have a symbolic discourse language that expresses any choice. But there's no mathematical derivation of the answer and there's no law of nature that forces a particular answer. I suppose there could be a "best answer given our biological nature". But as things advance, this won't be on solid ground either, as we increasingly manage to use technology to transcend the biology that evolution has delivered to us.

Still, we might argue, there's at least one constraint: we don't want a scheme where we'll go extinct—and where nothing will in the end exist. Even this is going to be a complicated thing to discuss, because we need to say what the "we" here is supposed to be: just how "evolved" relative to the current human condition can things be, and not consider "us" to have gone extinct?

But even independent of this, there's another issue: given any particular setup, computational irreducibility can make it in a sense irreducibly difficult to find out its consequences. And so in particular, given any specific optimization criterion (or constitution), there may be no finite procedure that will determine whether it allows for infinite survival, or whether in effect it implies civilization will "halt" and go extinct.

OK, so things are complicated. What can one actually do? For a little while there'll probably be the notion that AIs must ultimately have human owners, who must act according to certain principles, following the usual way human society operates. But realistically this won't last long.

Who would be responsible for a public-domain AI system that's spread across the internet? What happens when the bots it spawns start misbehaving on social media (yes, the notion that social media accounts are just for humans will soon look very "early 21st century")?

Of course, there's an important question of why AIs should "follow the rules" at all. After all, humans certainly don't always do that. It's worth remembering, though, that we humans are probably a particularly difficult case: after all, we're the product a multibillion-year process of natural selection, in which there's been a continual competitive struggle for survival. AIs are presumably coming into the world in very different circumstances, and without the same need for "brutish instincts". (Well, I can't help thinking of AIs from different companies or countries being imbued by their creators with certain brutish instincts, but that's surely not a necessary feature of AI existence.)

In the end, though, the best hope for getting AIs to "follow the rules" is probably by more or less the same mechanism that seems to maintain human society today: that following the rules is the way some kind of dynamic equilibrium is achieved. But if we can get the AIs to "follow the rules", we still have to define what the rules—the AI Constitution—should be.

And, of course, this is a hard problem, with no "right answer". But perhaps one approach is to see what's happened historically with humans. And one important and obvious thing is that there are different countries, with different laws and customs. So perhaps at the very least we have to expect that there'd be multiple AI Constitutions, not just one.

Even looking at countries today, an obvious question is how many there should be. Is there some easy way to say that—with technology as it exists, for example—7 billion people should be expected to organize themselves into about 200 countries?

It sounds a bit like asking how many planets the solar system should end up with. For a long time this was viewed as a "random fact of nature" (and widely used by philosophers as an example of something that, unlike 2+2=4, doesn't "have to be that way"). But particularly having seen so many exoplanet systems, it's become clear that our solar system actually pretty much has to have about the number of planets it does.

And maybe after we've seen the sociologies of enough video-game virtual worlds, we'll know something about how to "derive" the number of countries. But of course it's not at all clear that AI Constitutions should be divided anything like countries.

The physicality of humans has the convenient consequence that at least at some level one can divide the world geographically. But AIs don't need to have that kind of spatial locality. One can imagine some other schemes, of course. Like let's say one looks at the space of personalities and motivations, and finds clusters in it. Perhaps one could start to say "here's an AI Constitution for that cluster" and so on. Maybe the constitutions could fork, perhaps almost arbitrarily (a "Git-like model of society"). I don't know how things like this would ultimately work, but they seem more plausible than what amounts to a single, consensus, AI Constitution for everywhere and everyone.

There are so many issues, though. Like here's one. Let's assume AIs are the dominant power in our world. But let's assume that they successfully follow some constitution or constitutions that we've defined for them. Well, that's nice—but does it mean nothing can ever change in the world? I mean, just think if we were still all operating according to laws that had been set up 200 years ago: most of society has moved on since then, and wants different laws (or at least different interpretations) to reflect its principles.

But what if precise laws for AIs were burnt in around the year 2020, for all eternity? Well, one might say, real constitutions always have explicit clauses that allow for their own modification (in the US Constitution it's Article V). But looking at the actual constitutions of countries around the world isn't terribly encouraging. Some just say basically that the constitution can be changed if some supreme leader (a person) says so. Many say that the constitution can be changed through some democratic process—in effect by some sequence of majority or similar votes. And some basically define a bureaucratic process for change so complex that one wonders if it's formally undecidable whether it would ever come to a conclusion.

At first, the democratic scheme seems like an obvious winner. But it's fundamentally based on the concept that people are somehow easy to count (of course, one can argue about which people, etc.). But what happens when personhood gets more complicated? When, for example, there are in effect uploaded human consciousnesses, deeply intertwined with AIs? Well, one might say, there's always got to be some "indivisible person" involved. And yes, I can imagine little clumps of pineal gland cells that are maintained to define "a person", just like in the past they were thought to be the seat of the soul. But from the basic science I've done I think I can say for certain that none of this will ultimately work—because in the end the computational processes that define things just don't have this kind of indivisibility.

So what happens to "democracy" when there are no longer "people to count"? One can imagine all sorts of schemes, involving identifying the density of certain features in "people space". I suppose one can also imagine some kind of bizarre voting involving transfinite numbers of entities, in which perhaps the axiomatization of set theory has a key effect on the future of history.

It's an interesting question how to set up a constitution in which change is "burned in". There's a very simple example in bitcoin, where the protocol just defines by fiat that the value of mined bitcoin goes down every year. Of course, that setup is in a sense based on a model of the world—and in particular on something like Moore's Law and the apparent short-term predictability of technological development. But following the same general idea, one might starting thinking about a constitution that says "change 1% of the symbolic code in this every year". But then one's back to having to decide "which 1%?". Maybe it'd be based on usage, or observations of the world, or some machine-learning procedure. But whatever algorithm or meta-algorithm is involved, there's still at some point something that has to be defined once and for all.

Can one make a general theory of change? At first, this might seem hopeless. But in a sense exploring the computational universe of programs is like seeing a spectrum of all possible changes. And there's definitely some general science that can be done on such things. And maybe there's some setup—beyond just "fork whenever there could be a change"—that would let one have a constitution that appropriately allows for change, as well as changing the way one allows for change, and so on.

Making It Happen

OK, we've talked about some far-reaching and foundational issues. But what about the here and now? Well, I think the exciting thing is that 300 years after Gottfried Leibniz died, we're finally in a position to do what he dreamed of: to create a general symbolic discourse language, and to apply it to build a framework for computational law.

With the Wolfram Language we have the foundational symbolic system—as well as a lot of knowledge of the world—to start from. There's still plenty to do, but I think there's now a definite path forward. And it really helps that in addition to the abstract intellectual challenge of creating a symbolic discourse language, there's now also a definite target in mind: being able to set up practical systems for computational law.

It's not going to be easy. But I think the world is ready for it, and needs it. There are simple smart contracts already in things like bitcoin and Ethereum, but there's vastly more that can be done—and with a full symbolic discourse language the whole spectrum of activities covered by law becomes potentially accessible to structured computation. It's going to lead to all sorts of both practical and conceptual advances. And it's going to enable new legal, commercial and societal structures—in which, among other things, computers are drawn still further into the conduct of human affairs.

I think it's also going to be critical in defining the overall framework for AIs in the future. What ethics, and what principles, should they follow? How do we communicate these to them? For ourselves and for the AIs we need a way to formulate what we want. And for that we need a symbolic discourse language. Leibniz had the right idea, but 300 years too early. Now in our time I'm hoping we're finally going to get to build for real what he only imagined. And in doing so we're going to take yet another big step forward in harnessing the power of the computational paradigm.

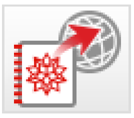
Posted in: Artificial Intelligence, Computational Thinking, Future Perspectives, Other

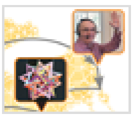
---

Join the discussion

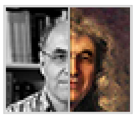
+ 10 comments

Related Writings


The New World of Notebook Publishing
October 24, 2019


Just Published: *Adventures of a Computational Explorer*
October 16, 2019

Testifying at the Senate about A.I.-Selected Content on the Internet
June 25, 2019


A Few Thoughts about Deep Fakes
June 12, 2019

Popular Categories

Artificial Intelligence	Mathematica
Big Picture	Mathematics
Companies and Business	New Kind of Science
Computational Science	New Technology
Computational Thinking	Personal Analytics
Data Science	Physical Science
Education	Software Design
Future Perspectives	Wolfram|Alpha
Historical Perspectives	Wolfram|One
Life and Times	Wolfram Language
Life Science	Other

Writings by Year

2019 | 2018 | 2017 | 2016 | 2015 | 2014 | 2013 | 2012 | 2011 | 2010 | 2009 | 2008 | 2007 | 2006 | 2004 | 2003 | All

© Stephen Wolfram, LLC | Terms | RSS