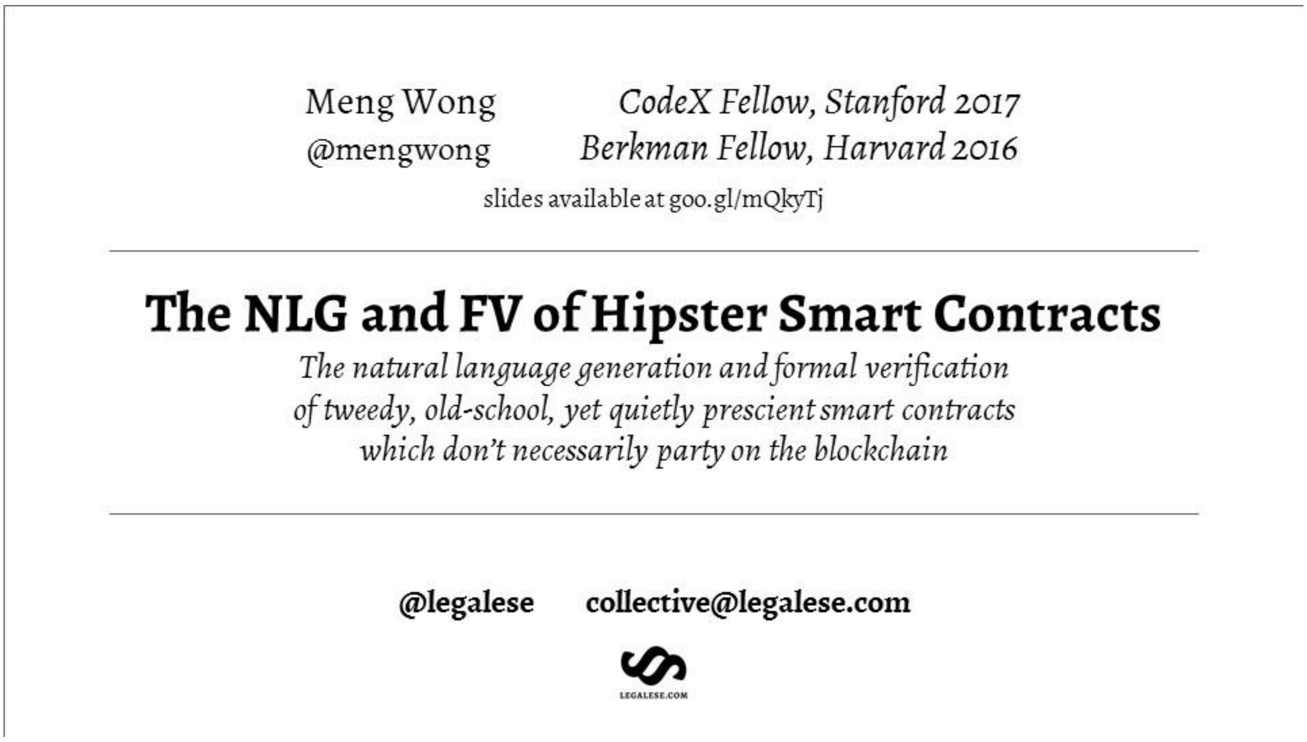

Meng Wong CodeX Fellow, Stanford 2017
@mengwong Berkman Fellow, Harvard 2016
slides available at goo.gl/mQkyTj

The NLG and FV of Hipster Smart Contracts

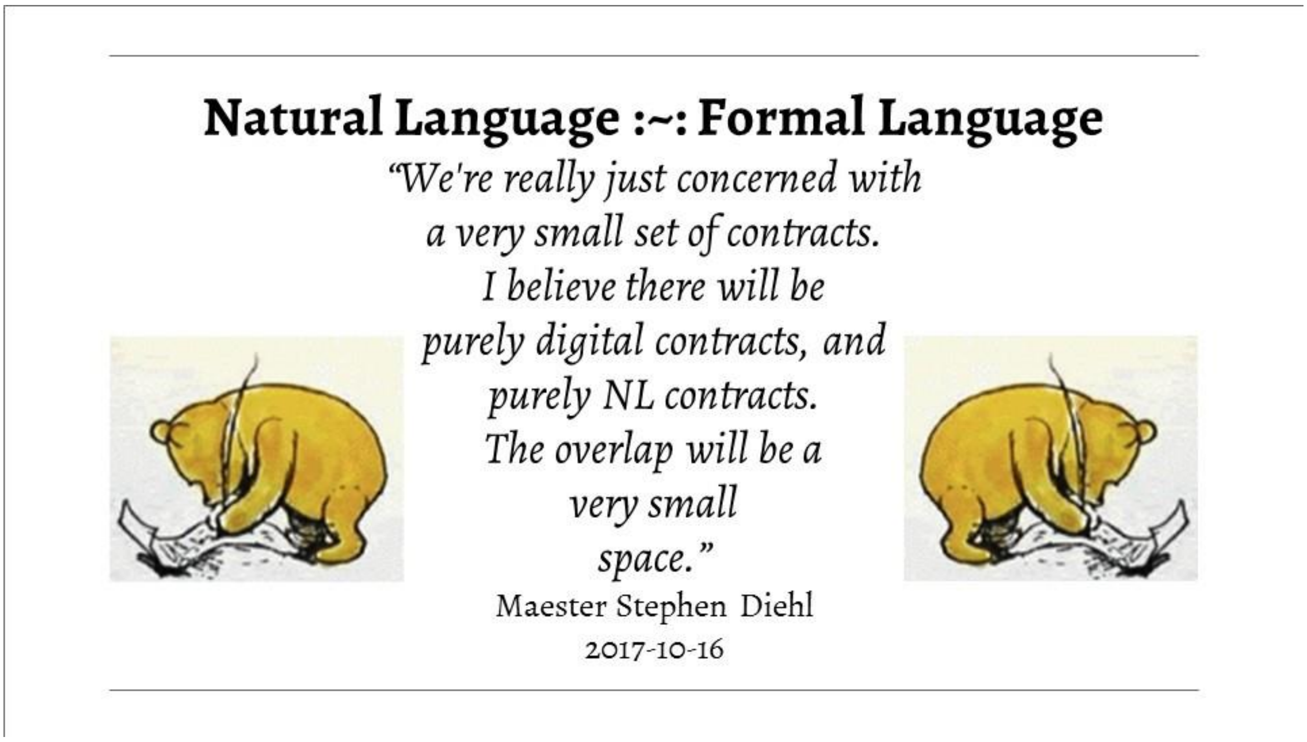
The natural language generation and formal verification of tweedy, old-school, yet quietly prescient smart contracts which don't necessarily party on the blockchain

@legalese collective@legalese.com

LEGALESE.COM

My name is Meng, and I'm going to give you a quick tour of what I call hipster smart contracts. These are contracts which were already trying to be smart even before blockchain was invented. [laughter]

I spend a lot of time around academic hipsters, because I am currently a Fellow at the CODEX Center for Legal Informatics at Stanford, where we have an open source, open standard project to develop a programming language for contracts, very similar to some of the other efforts that we've seen. We are a little contrarian in that we are not completely obsessed with blockchain, which is what we call computable contracts, not smart contracts. Last month about 50 people came together at Stanford to talk about use cases for computable contracts so that we don't end up with a solution looking for a problem, and that effort is underway. Because it's an academic effort, I except it will bear fruit anywhere from two to three years from now, but maybe sooner.



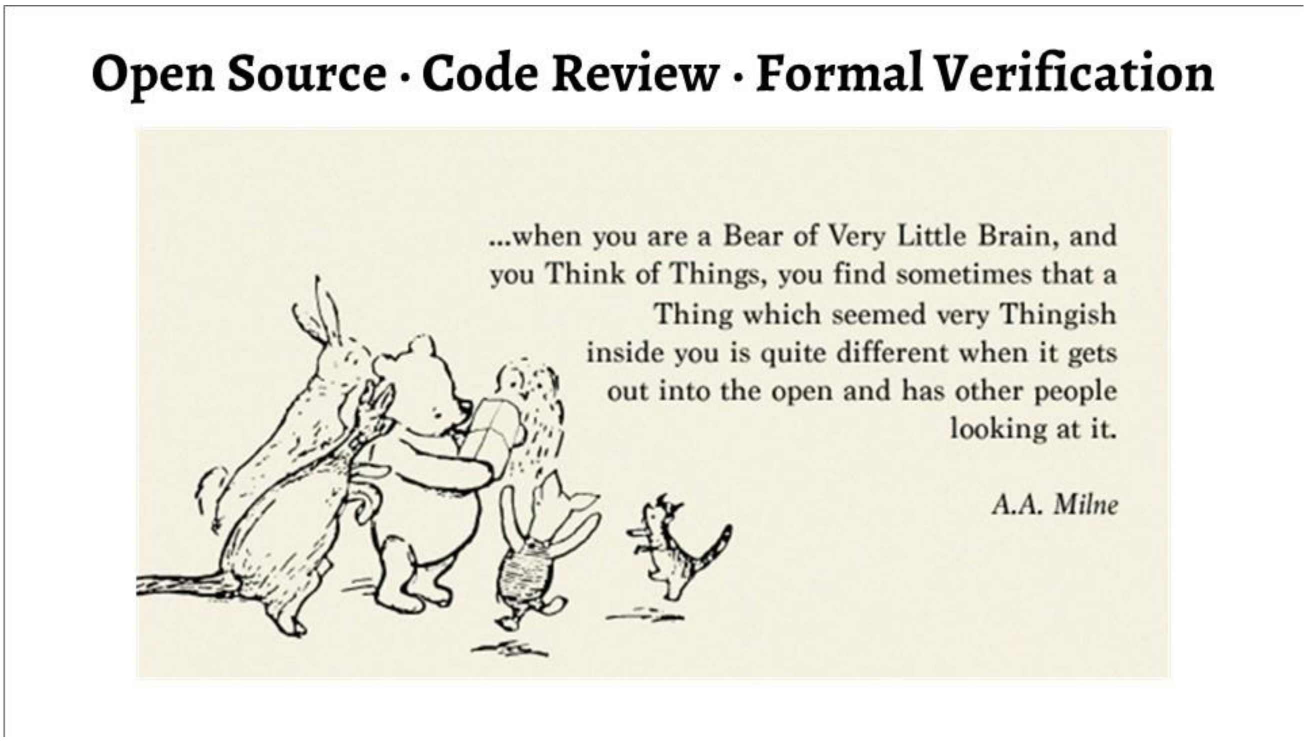
Natural Language :~: Formal Language

"We're really just concerned with a very small set of contracts. I believe there will be purely digital contracts, and purely NL contracts. The overlap will be a very small space."

Maester Stephen Diehl

2017-10-16

I want to build on some of the things that were said this morning by Christopher Brewster and by Stephen Diehl. This morning we heard from Maester Stephen, who said that the overlap between natural language and digital contracts might be very small. If that turns out to be true, I will be okay with that, because if you are a bear of very little brain, then a small overlap is just the right size. In fact, Winnie the Pooh did say *this.*

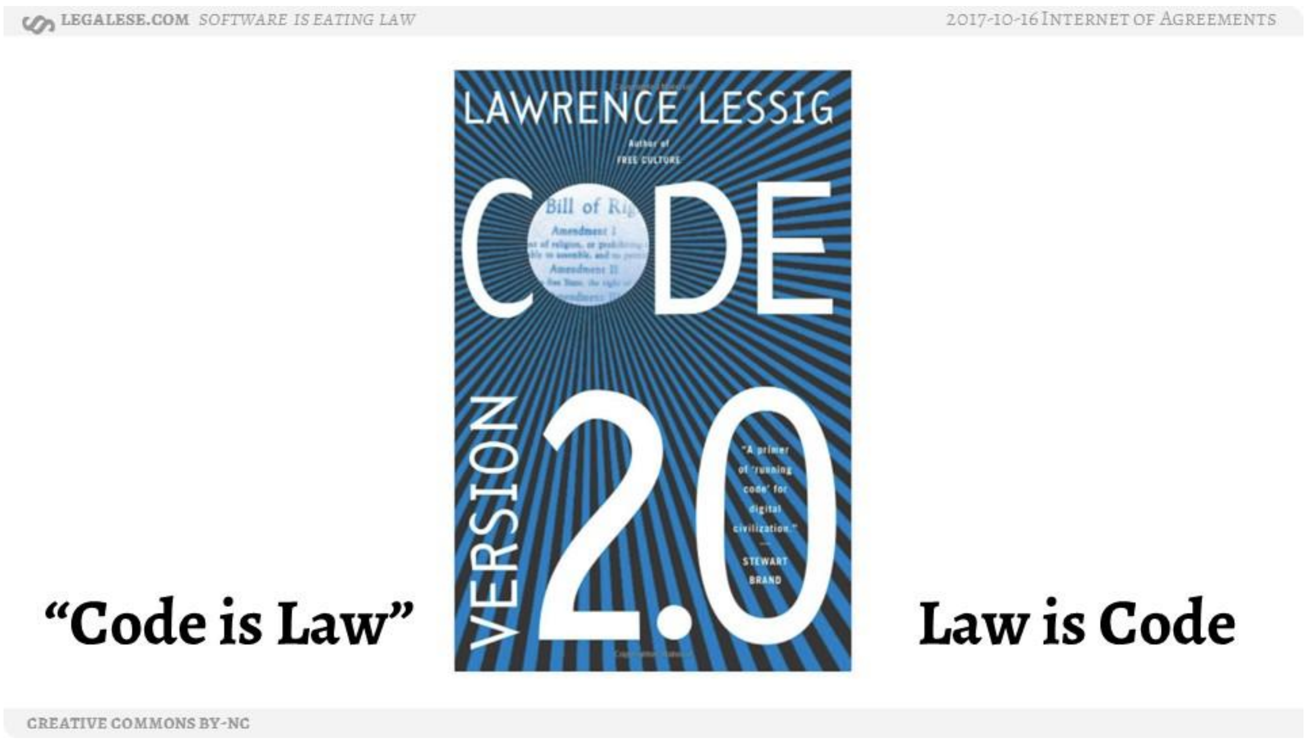


Open Source · Code Review · Formal Verification

...when you are a Bear of Very Little Brain, and you Think of Things, you find sometimes that a Thing which seemed very Thingish inside you is quite different when it gets out into the open and has other people looking at it.

A.A. Milne

This suggests to me that almost a century ago Winnie the Pooh had already invented such terribly modern ideas as open source, coder views, formal verification and other such processes. [laughter]


LEGALESE.COM *SOFTWARE IS EATING LAW* 2017-10-16 INTERNET OF AGREEMENTS

"Code is Law" **Law is Code**

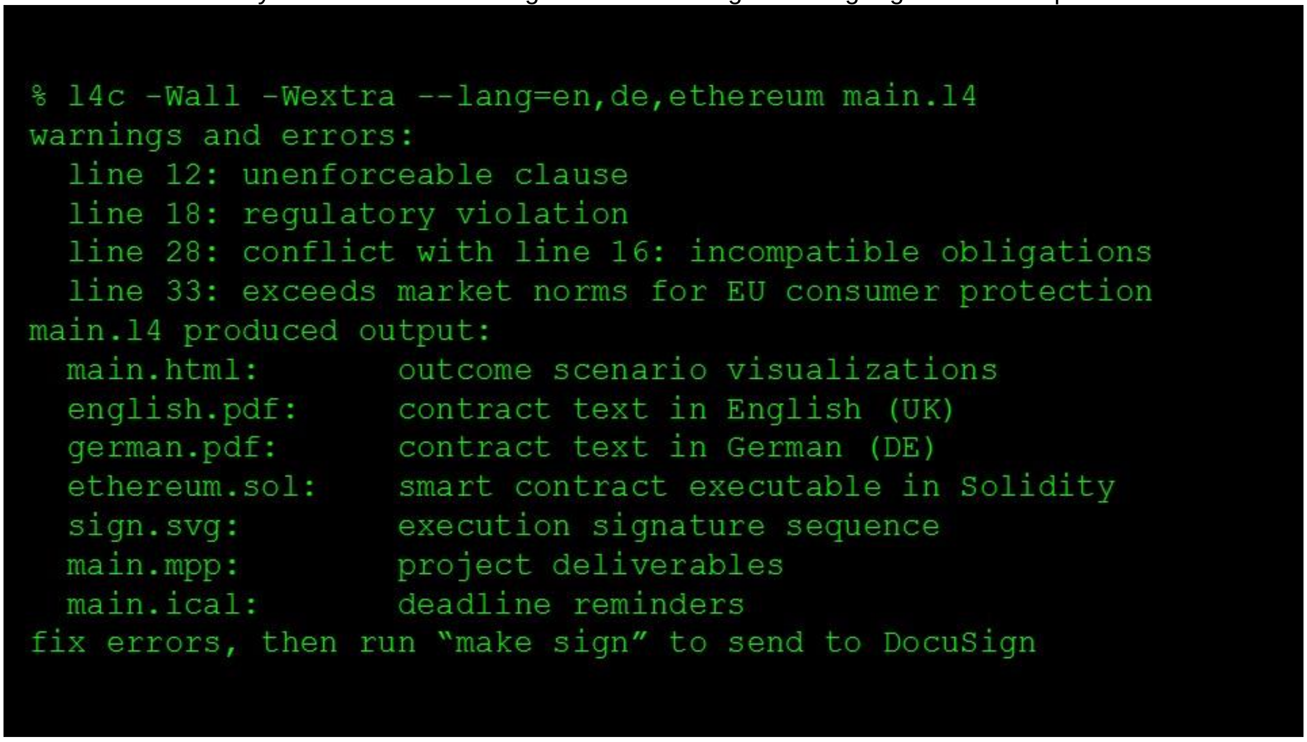
LAWRENCE LESSIG
Author of FREE CULTURE
CODE
VERSION 2.0
"A primer of 'running code' for digital civilization." — STEWART BRAND

CREATIVE COMMONS BY-NC

But coming back to the overlap between code and law, the inspiration for computable contracts comes from Lawrence Lessig saying that code is law. The idea that law is code became very real for me a few years ago when I was working on investment agreements like *this one*. I was just reading documentation like this all day long, and I thought, speaking as a programmer, this time it really wants to get out of Microsoft Word and into implementations and syntax highlighting... This is what that agreement really wants to be when it grows up: it wants to be a program. [laughter] Once you have a program, the cool thing about this is that you should be able to take 10 lines of code and compile it back out to 10 pages of PDF, and that's called isomorphism. That's how software works, and I think that's how contracts should work. Even the UK Law Society Gazette agrees: they call contract lawyers copy & paste monkeys. I can't tell you how many times I've asked my lawyer, "Can I please have some paperwork?" and they have given it to me with the previous client's name still in it. [laughter] So no wonder JP Morgan went and built themselves an app to replace human lawyers with just some software.

Now, of course the lawyers will protest, they will say, "We add value," but if you unbundle that value and examine it for a moment, from a linguistic perspective we're moving up the curve, from syntax to semantics to pragmatics, from "What does it say?" to "What does it mean?" to "What does it mean for me?"

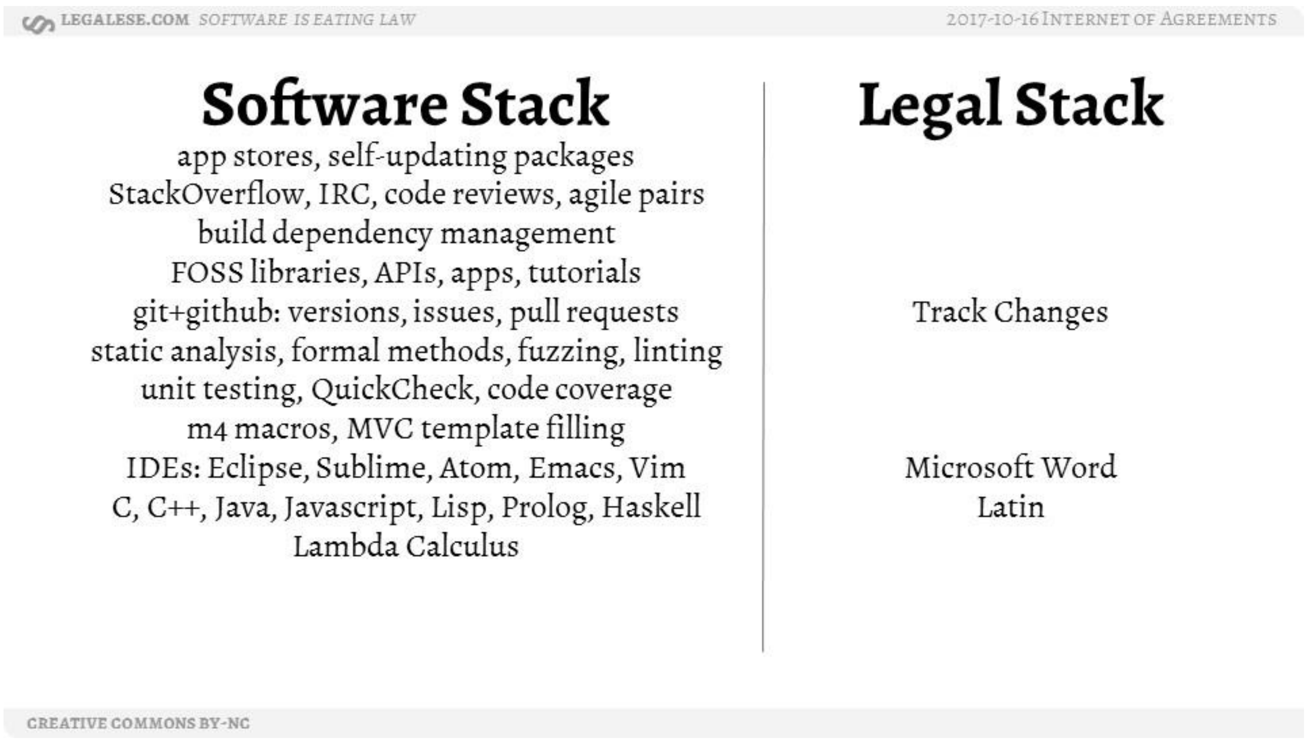
and that is where lawyers are used to adding value. But imagine a language and a compiler that can do *this.*



```
% l4c -Wall -Wextra --lang=en,de,ethereum main.l4
warnings and errors:
  line 12: unenforceable clause
  line 18: regulatory violation
  line 28: conflict with line 16: incompatible obligations
  line 33: exceeds market norms for EU consumer protection
main.l4 produced output:
  main.html:       outcome scenario visualizations
  english.pdf:     contract text in English (UK)
  german.pdf:      contract text in German (DE)
  ethereum.sol:    smart contract executable in Solidity
  sign.svg:        execution signature sequence
  main.mpp:        project deliverables
  main.ical:       deadline reminders
fix errors, then run "make sign" to send to DocuSign
```

The language is called L4, here's a compiler for you and we've turned on all the warnings, and we want to output to English, German and Ethereum. The compiler tells you "Here are some things that are wrong with your contract: you've got an unenforceable clause, you've got a conflict that exceeds the market norms," it knows about the regulation, and so it refuses to compile your contract until you fix these bugs. And when it does compile your contract, you get English, German, Solidity, you get your make file of sequential dependencies and it shoves it into a signature bracket. This is what you want, and this is kind of what lawyers have been doing and they haven't been very happy doing this, so I think the junior lawyers that do this today are okay with changing it. If you're familiar with programming, you know the if-then-else construct; if you know SQL, then you're familiar with the SELECT statement. Now, our contribution the lexicon of the software will be the "shall" clause... [laughter] It's possible that this will do for the legal industry what SQL did for the database industry.

So the novel element here is the "shall". We borrowed a few other words from law, so if you're a legal engineer, you get to say "shall", "hence" and "lest" instead of "if", "then" and "else" so you can add fees. [laughter] But basically, you can think of a contract as just a big graph of these clauses and you connect them together, and if the party does the thing then control passes to this other clause, and if they don't do the thing then it passes to this other clause, and that's what a contract is. I'm starting with a language and a compiler, because they form the foundation for a whole stack of technologies for legal engineering. I used the word "stack", and let me explain what I mean when I say the word "stack".

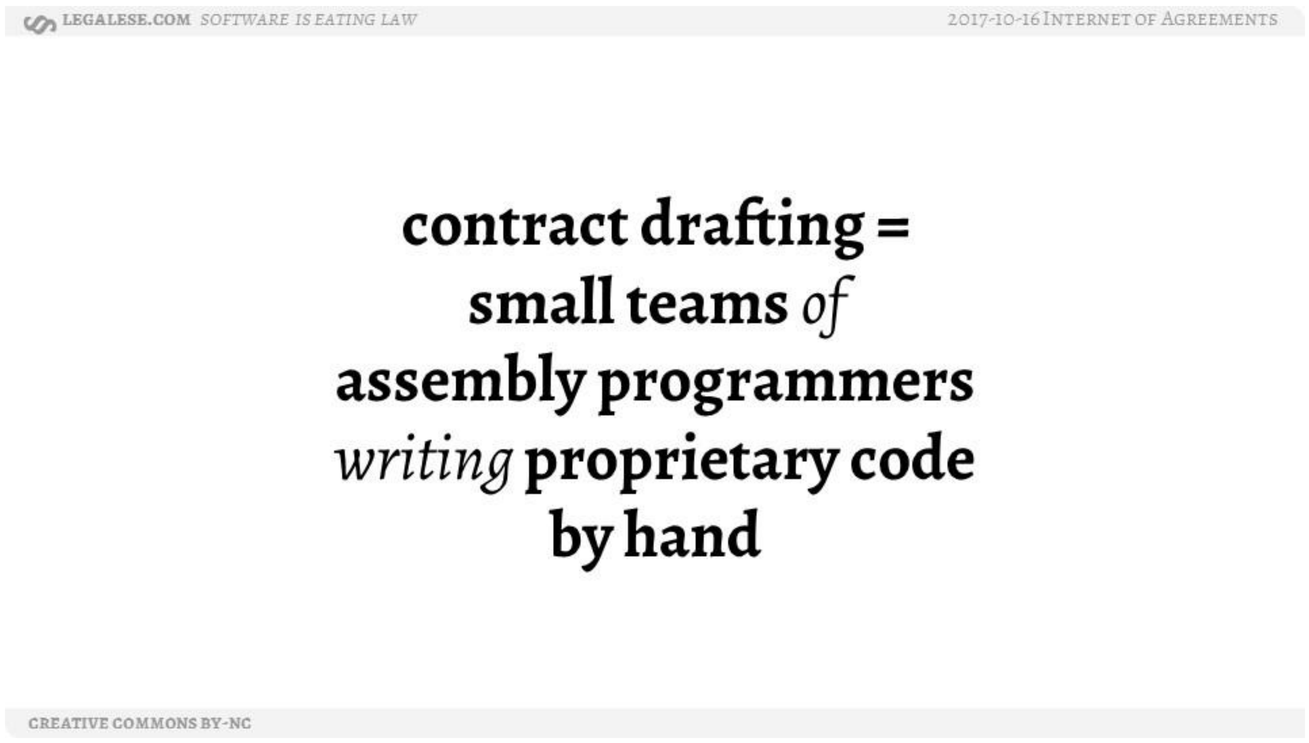


LEGALESE.COM *SOFTWARE IS EATING LAW* 2017-10-16 INTERNET OF AGREEMENTS

Software Stack	Legal Stack
app stores, self-updating packages	
StackOverflow, IRC, code reviews, agile pairs	
build dependency management	
FOSS libraries, APIs, apps, tutorials	
git+github: versions, issues, pull requests	Track Changes
static analysis, formal methods, fuzzing, linting	
unit testing, QuickCheck, code coverage	
m4 macros, MVC template filling	
IDEs: Eclipse, Sublime, Atom, Emacs, Vim	Microsoft Word
C, C++, Java, Javascript, Lisp, Prolog, Haskell	Latin
Lambda Calculus	

CREATIVE COMMONS BY-NC

Over the last 50 years, software engineers... Who here is a software engineer? Okay, good. Members of our tribe, we’ve gone and developed a massive suite of tools just to make our lives easier, and these tools help us do our job. At the bottom we’ve got languages that are basically based on mathematics, the land of calculus, and at the top we’ve got user-facing applications that show up at the app store, and in the middle we’ve got all kinds of tooling and infrastructure that give you things like type checking and unit testing and version control, and if you’re a programmer you should recognise these ideas. Now, I am told that in the legal industry, which is structurally very similar to the software industry in some ways... What does the legal stack look like? I’ve been told that in most law firms the most advanced technology they have is track changes. [laughter] I’m not trying to make fun of lawyers here, some of my best friends are lawyers, but I just like to think that they could have been programmers if they just made some better life choices. [laughter] So every one of these white spaces is an opportunity, and it’s going to get filled in the next few years I think.



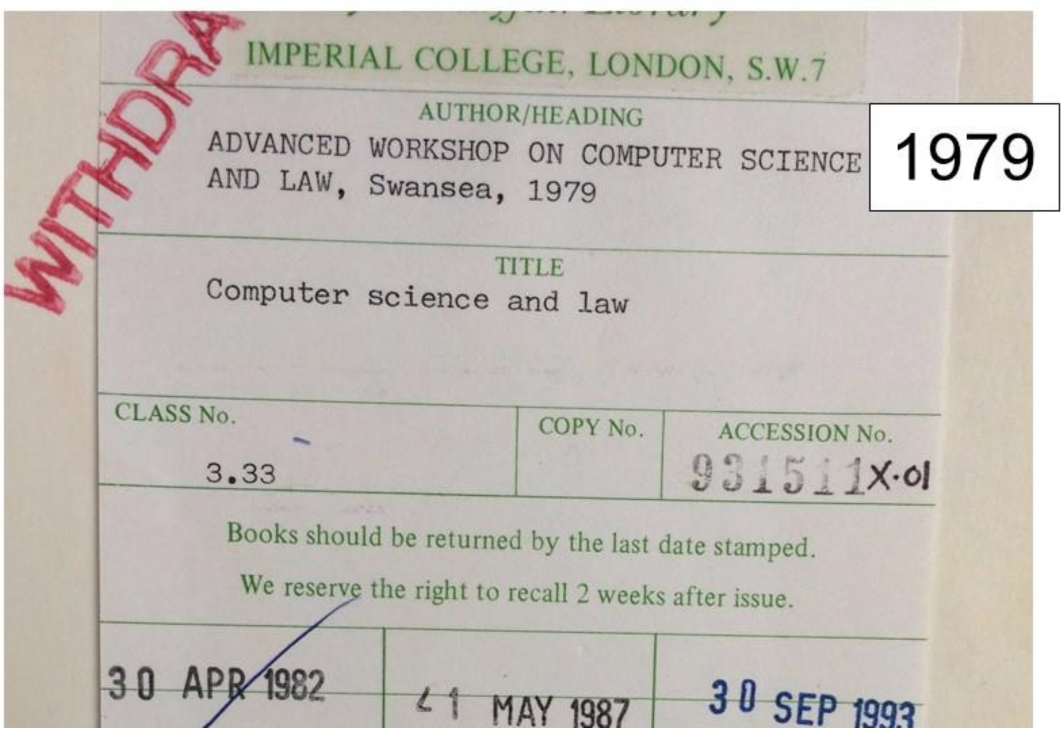
LEGALESE.COM *SOFTWARE IS EATING LAW* 2017-10-16 INTERNET OF AGREEMENTS

contract drafting =
small teams *of*
assembly programmers
writing **proprietary code**
by hand

CREATIVE COMMONS BY-NC

So the insight here is that really when you’re drafting a contract and you’re programming some software, you’re doing the same thing, but they’re sort of where we were back in the 1970s, you’re writing assembly code by hand. 50 years ago software was proprietary, there were no open languages, there was no open source and there was no Git; you’d have to hire a team of programmers to do it by hand from scratch, and it would start at $100,000. That’s where most big law firms are today.

But if you go back, I’ve spent the last two years in academia studying the history of this, in 1960 we already had the idea that we should be able to move thinking into a machine. This is generations ago people were doing this, in 1957 already we had the idea of writing legal documents using formal logic, and they don’t teach this in law school.



WITHDRA

IMPERIAL COLLEGE, LONDON, S.W.7

AUTHOR/HEADING

ADVANCED WORKSHOP ON COMPUTER SCIENCE AND LAW, Swansea, 1979

1979

TITLE

Computer science and law

CLASS No.	COPY No.	ACCESSION No.
3.33		931511X·01

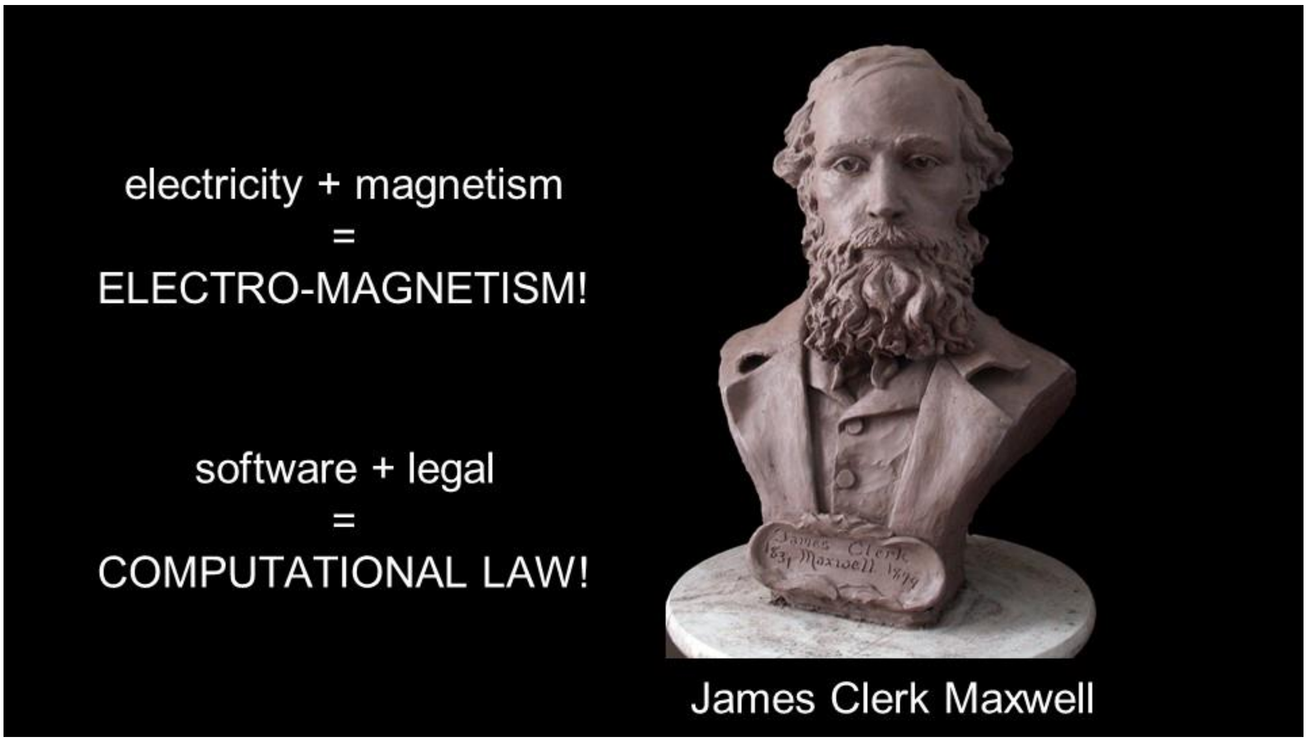
Books should be returned by the last date stamped.

We reserve the right to recall 2 weeks after issue.

30 APR 1982	21 MAY 1987	30 SEP 1993

I came across this really interesting book in the 1970s, it was a DX session from a library, Computer Science and Law. This is one of those books where every single chapter in this book has turned into its whole segment of the legal industry today. For example, this project in 1977 tried to make tax law computable, and that was basically the granddaddy of TurboTax. This chapter talking about modelling legal rules, that's what powers the expert systems and the chatbots of today. This chapter was the granddaddy of a whole industry which we call document assembly. So the whole idea of a smart contract does predate blockchain, in 1997 Nick Szabo was talking about this, he posted on his blog about smart contracts. This was pretty pioneering of him, because blogs haven't even been invented yet and smart contracts haven't been invented, and here he was blogging about smart contracts. [laughter] Fantastic: in 2002 how to formalise smart contracts.

So the whole academic tradition of research into the formalisation of smart contracts has been going on uninterrupted for years on this. There are papers that you can read about how this works for contracts, how this works for law. I've been reading a stack of papers about *this* thick on how to do conflict analysis, how to extract the formalisation from natural language. There are more papers than you can imagine. That's why we're getting into the Semantic Web stuff, this is where you have books on ontologies. Every once in a while I find something that gets me really excited, and *this* is the formal of semantics for contract language that was the subject of a PhD thesis from a few years ago, and when I saw this, I said, "Aha, I have found it, I have found the missing link!" Because this is the adaptor that lets you plug the legal industry into the software industry, connect them up, thanks to this formalisation, which represents basically one of those clauses I showed you. Once you do that, you can turn contracts into diagrams, you can turn diagrams into contracts. We call this a visual isomorphism, where you go from the program to the flowchart and the flowchart to the program. There's a whole history around this, and if you remember the UML [unified modelling language] days of SBVR [semantics of business vocabulary and business rules], BPMN [business process modelling and notation] DMN [decision model and notation]... You know what I'm talking about. [laughter]

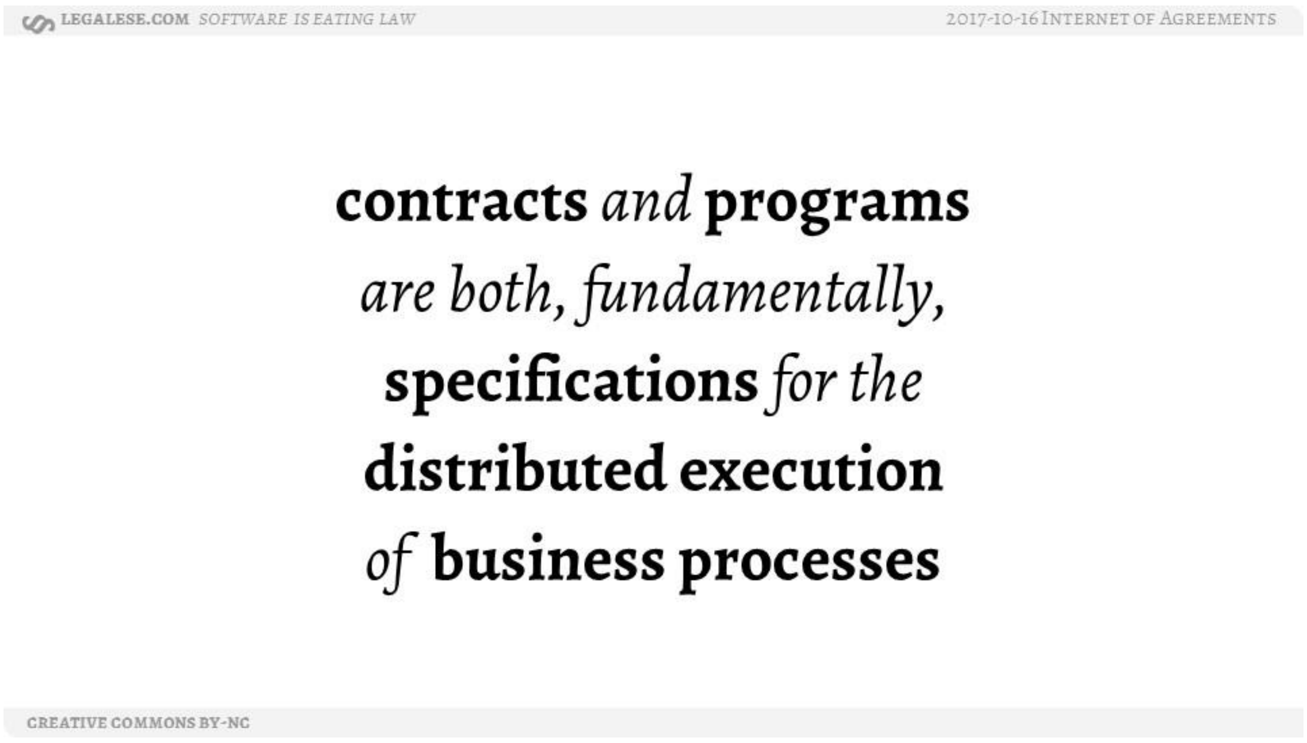

electricity + magnetism
=
ELECTRO-MAGNETISM!

software + legal
=
COMPUTATIONAL LAW!

James Clerk 1831 Maxwell 1879

James Clerk Maxwell

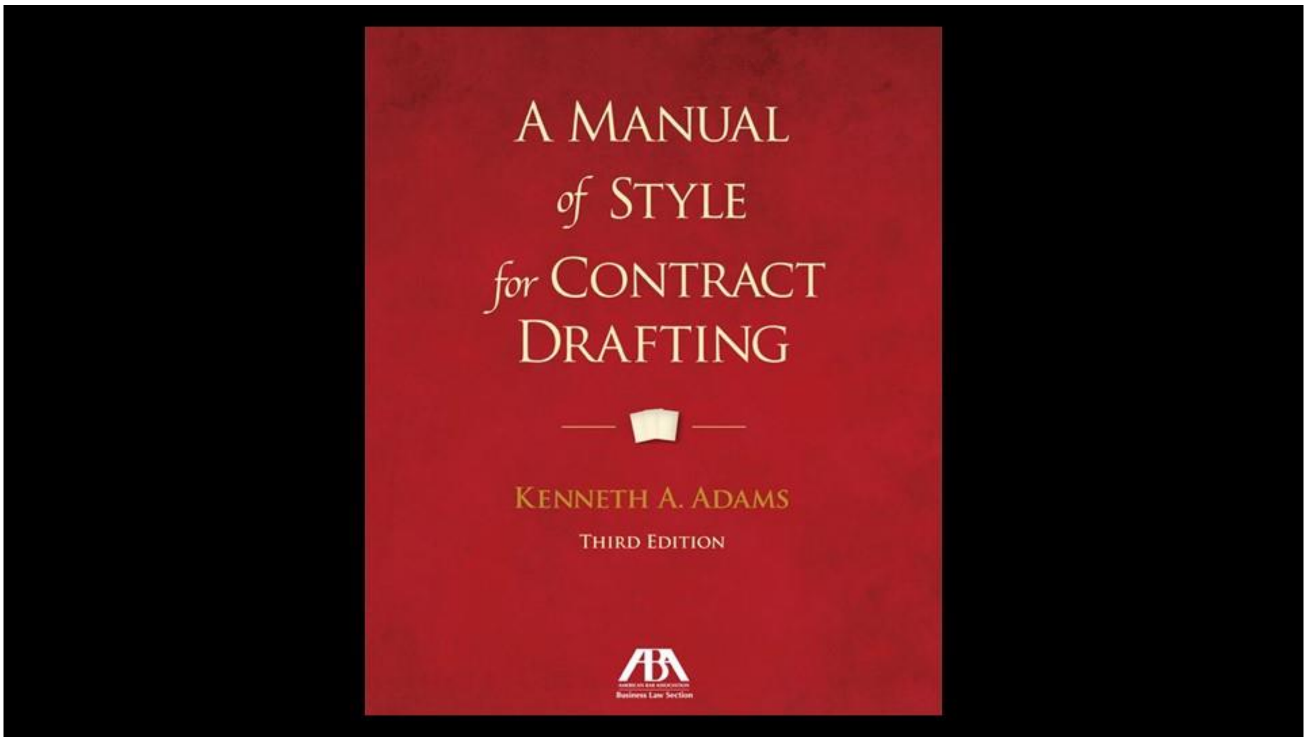
So, the big idea here, since we're in the UK, if James Clerk Maxwell were with us today, he would say, "This is a game I know how to play. You take two things that everybody thinks are different, and you think very hard and they become one thing," and that's called a unification. When it comes to law, all we have to do is think very hard and we should get another unification: computational law.


LEGALESE.COM *SOFTWARE IS EATING LAW* 2017-10-16 INTERNET OF AGREEMENTS

contracts *and* **programs**
are both, fundamentally,
specifications *for the*
distributed execution
of **business processes**

CREATIVE COMMONS BY-NC

The reason this is possible is because contracts and programs are fundamentally the same thing, they are two sides of the same coin: they are both specifications for the distributed execution of business processes, that's what makes it possible for us to do this. I'm going to give you a couple of very technical examples that build on what Stephen and Christopher were talking about earlier today. Shall I spend a few minutes getting very technical, yes? [crowd approves]



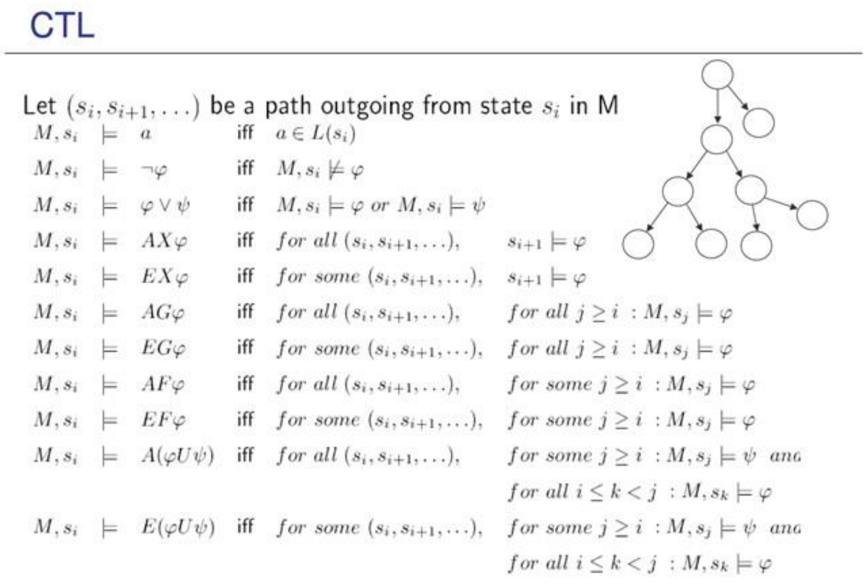
A MANUAL of STYLE for CONTRACT DRAFTING

KENNETH A. ADAMS

THIRD EDITION

American Bar Association Business Law Section

Ken Adams, this guy is a lawyer who literally wrote the book on contract drafting. A few months ago he posted *this* on Twitter, asking a question for the new edition of this book: how do you clarify the two senses of the word shall? In the first case, what if it's ongoing? "Acme shall keep the information confidential forever," and in the other case "Acme shall pay the purchase price," presumably once and not every 10 minutes. When I saw this, I knew that we were going to win, because he is asking the wrong people. He is asking linguists when he should be asking the geeks, because we've got the math to answer this question. [laughter] I'm going to show you exactly what the logic is. Stephen was talking about LTL and CTL earlier. This is the stuff that hardware designers used to making so that their chips don't have bugs. If you ship a CPU with a bug, the recall could cost millions; when Intel had the FDIV Pentium bug, that cost $475 million. You have to get it right the first time, just like contracts, so let's use the same technology.

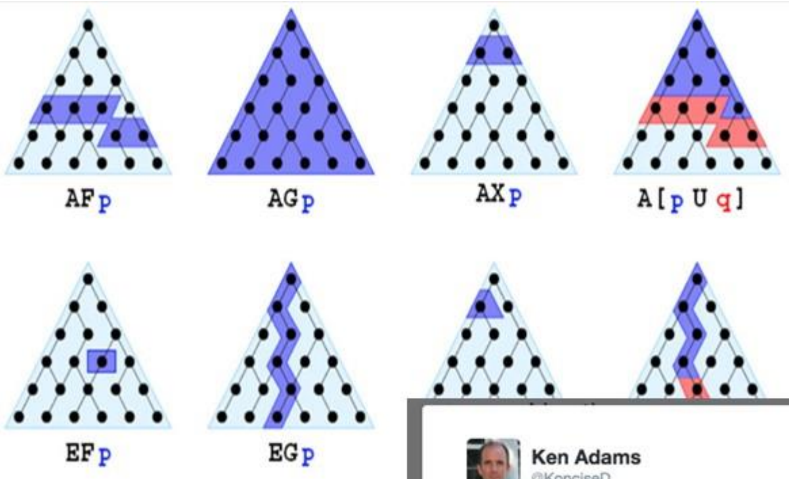


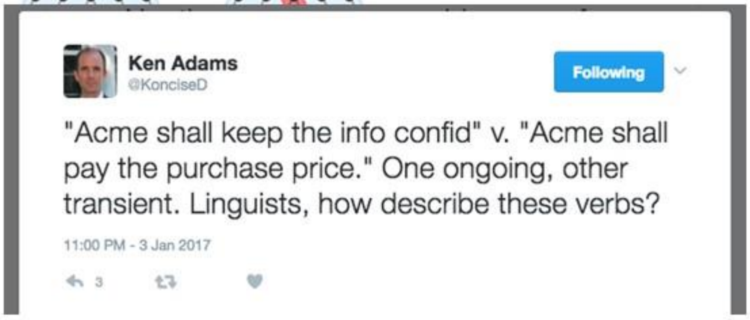
CTL

Let $(s_i, s_{i+1}, \ldots)$ be a path outgoing from state s_i in M

M, s_i	$\models$	a	iff	$a \in L(s_i)$	
M, s_i	$\models$	$\neg\varphi$	iff	$M, s_i \not\models \varphi$	
M, s_i	$\models$	$\varphi \vee \psi$	iff	$M, s_i \models \varphi$ *or* $M, s_i \models \psi$	
M, s_i	$\models$	$AX\varphi$	iff	*for all* $(s_i, s_{i+1}, \ldots),$	$s_{i+1} \models \varphi$
M, s_i	$\models$	$EX\varphi$	iff	*for some* $(s_i, s_{i+1}, \ldots),$	$s_{i+1} \models \varphi$
M, s_i	$\models$	$AG\varphi$	iff	*for all* $(s_i, s_{i+1}, \ldots),$	*for all* $j \geq i\ : M, s_j \models \varphi$
M, s_i	$\models$	$EG\varphi$	iff	*for some* $(s_i, s_{i+1}, \ldots),$	*for all* $j \geq i\ : M, s_j \models \varphi$
M, s_i	$\models$	$AF\varphi$	iff	*for all* $(s_i, s_{i+1}, \ldots),$	*for some* $j \geq i\ : M, s_j \models \varphi$
M, s_i	$\models$	$EF\varphi$	iff	*for some* $(s_i, s_{i+1}, \ldots),$	*for some* $j \geq i\ : M, s_j \models \varphi$
M, s_i	$\models$	$A(\varphi U \psi)$	iff	*for all* $(s_i, s_{i+1}, \ldots),$	*for some* $j \geq i\ : M, s_j \models \psi$ *an*
					for all $i \leq k < j\ : M, s_k \models \varphi$
M, s_i	$\models$	$E(\varphi U \psi)$	iff	*for some* $(s_i, s_{i+1}, \ldots),$	*for some* $j \geq i\ : M, s_j \models \psi$ *an*
					for all $i \leq k < j\ : M, s_k \models \varphi$

I’m not going to teach you CTL right now, I’m just going to show you that diagram *on the right*, that top circle represents the current state and the lower circles represent future states. CTL is an algebra that gives us a language to say something will be true for every possible future, or something will be true for some possible future, or something will eventually be true in every possible future. I’m not going to go through all the different possibilities, but the point is that mathematicians and computer scientists and logicians have a very precise way to talk about time, and this precision is something that is sort of lacking in how lawyers are trained I think, and that leads to drafting errors, so these kinds of terms are exactly what you need in contracts.

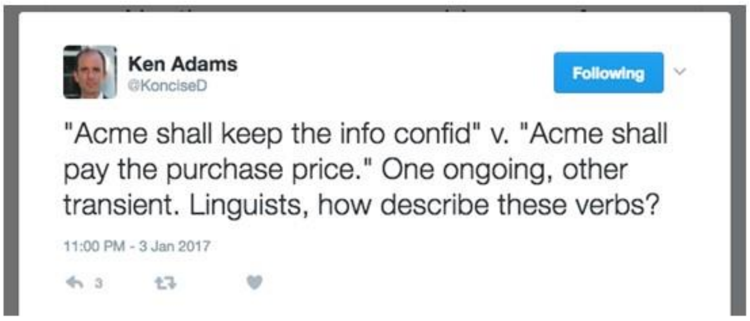



Ken Adams
@KonciseD
Following

"Acme shall keep the info confid" v. "Acme shall pay the purchase price." One ongoing, other transient. Linguists, how describe these verbs?

11:00 PM - 3 Jan 2017

If you take these possibilities, we now have the equipment needed to answer this question. In the first case, Acme shall keep the information confidential forever... Which one of *these* is that statement? AG. As opposed to Acme shall pay the purchase price once in every future but not after that. That's AF. So AF versus AG.


Ken Adams
@KonciseD
Following

"Acme shall keep the info confid" v. "Acme shall pay the purchase price." One ongoing, other transient. Linguists, how describe these verbs?

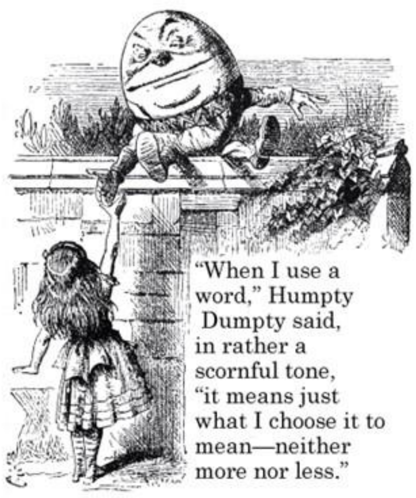
11:00 PM - 3 Jan 2017

Acme shall keep the info confidential:	**AG** (keep_confidential info)
Acme shall pay the purchase price:	**AF** (pay purchase_price)

Now we have a way of formalising that in a way that the computer can read, and that allows to do automated reasoning.

LEGALESE.COM *SOFTWARE IS EATING LAW* 2017-10-16 INTERNET OF AGREEMENTS

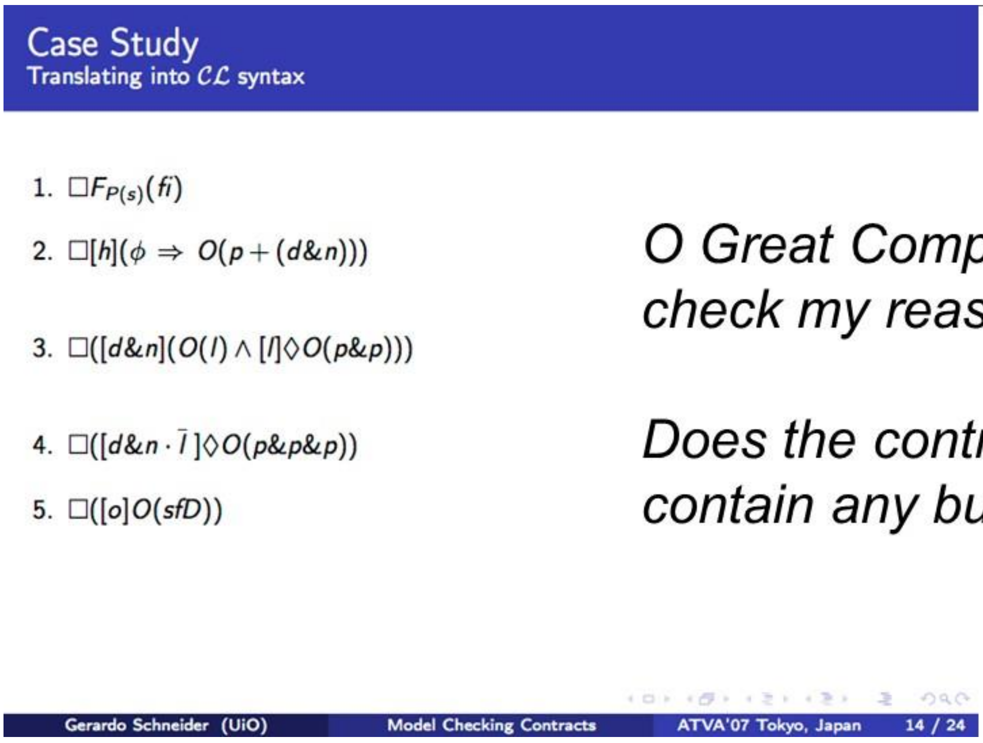
AUTOMATED BUG-FINDING



"When I use a word," Humpty Dumpty said, in rather a scornful tone, "it means just what I choose it to mean—neither more nor less."

CREATIVE COMMONS BY-NC

We've seen earlier the importance of ontologies and formal semantics. Humpty Dumpty says, "It means just what I choose it to mean." But if you are essentially solipsistic and subjective, then it's only a matter of time before you have a great fall: you trigger a runtime exception in your contract. In contracts, what do you call a runtime exception? It's called a lawsuit. [laughter] You want to save that lawsuit, you want to detect these bugs at compile time and not at runtime, and that's why you hire an expert swordsman to come poke holes in your code before you run it. One way to do that is called model checking, this is one method of doing formal verification, this formal method, and this is some work that was done about 10 years ago, where if you take a contract... You don't have to read this, but this is basically an SLA contract that you see in ISPs and telcos: what you do is you take every line of this contract and you translate it into the kind of algebra that we just looked, and this is just a slightly fancier version of what I just showed you: you take line by line, boom-boom-boom, and now you've got something that is much clearer to a computer.



Case Study
Translating into $\mathcal{CL}$ syntax

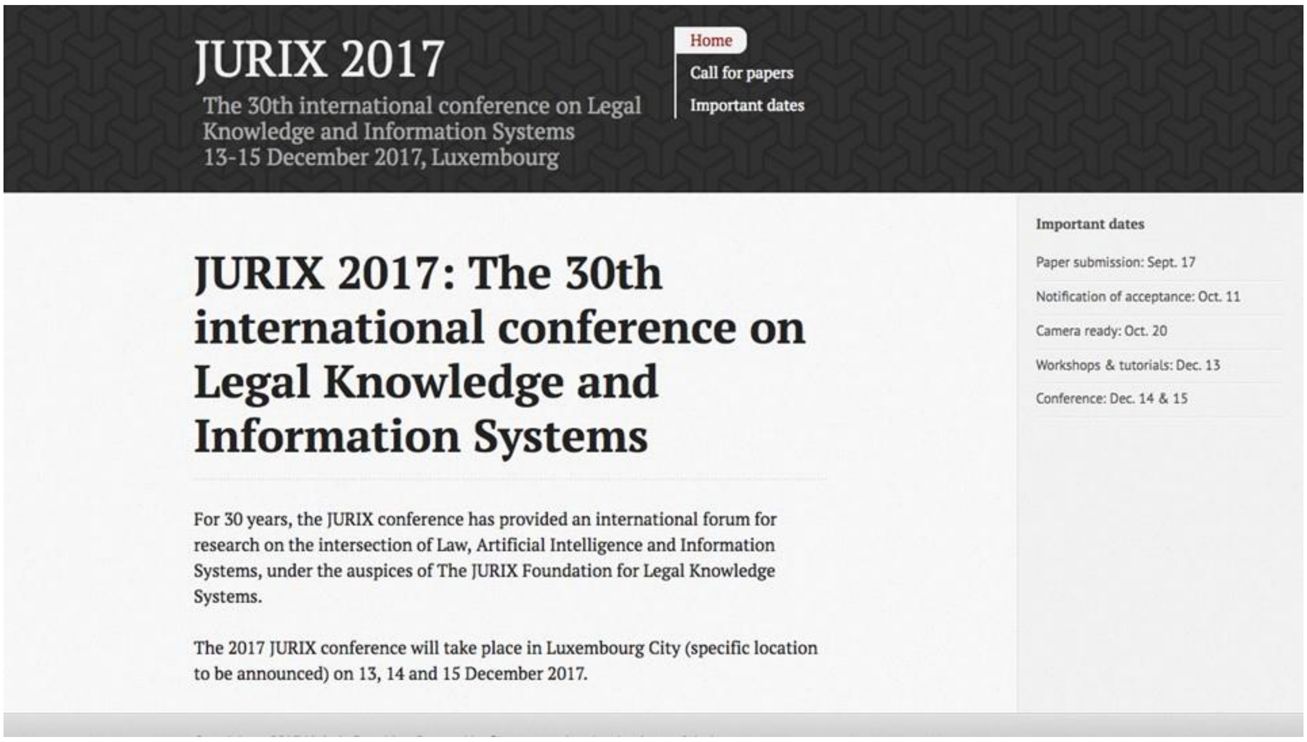
1. $\Box F_{P(s)}(fi)$
2. $\Box[h](\phi \Rightarrow O(p + (d\&n)))$
3. $\Box([d\&n](O(l) \wedge [l]\Diamond O(p\&p)))$
4. $\Box([d\&n \cdot \bar{l}\,]\Diamond O(p\&p\&p))$
5. $\Box([o]O(sfD))$

O Great Computer, check my reasoning.

Does the contract contain any bugs?

Gerardo Schneider (UiO) Model Checking Contracts ATVA'07 Tokyo, Japan 14 / 24

Because now you can say to the computer, “Here’s my contract, and here’s what I want the contract to do, here’s what it actually does. Is this correct?” We feed these things into the model checker, and the model checker says, “If there’s a bug, it will give you back a possible execution trace that leads to that bug. Here’s a scenario that you don’t want to happen, and here’s how we can get to that scenario.” This is exactly the kind of thing that lawyers get paid to do, only we can now do it in software for 10 cents an hour.


JURIX 2017

The 30th international conference on Legal Knowledge and Information Systems
13-15 December 2017, Luxembourg

Home
Call for papers
Important dates

JURIX 2017: The 30th international conference on Legal Knowledge and Information Systems

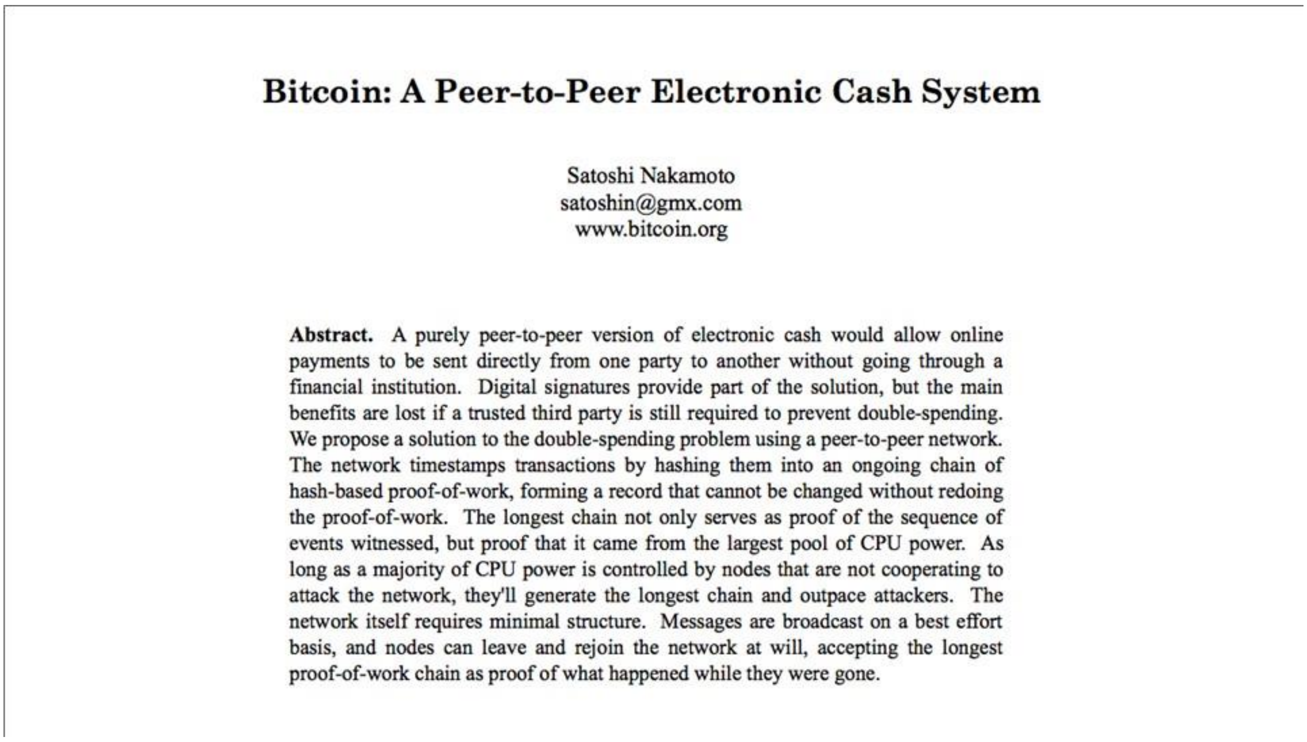
For 30 years, the JURIX conference has provided an international forum for research on the intersection of Law, Artificial Intelligence and Information Systems, under the auspices of The JURIX Foundation for Legal Knowledge Systems.

The 2017 JURIX conference will take place in Luxembourg City (specific location to be announced) on 13, 14 and 15 December 2017.

Important dates

Paper submission: Sept. 17
Notification of acceptance: Oct. 11
Camera ready: Oct. 20
Workshops & tutorials: Dec. 13
Conference: Dec. 14 & 15

So, what is my point? Even before the current wave of legal tech and blockchain, there has been a huge amount of academic research in this. This is the 30th year that *this* conference has been going on in legal knowledge and information systems. Right here in London in June was the ICAIL Conference, and the they only have this every second year, so this is the 32nd year that they've done this. *This* was going in Malta 10 years ago, formal languages and contract-oriented software. So there is thinking about this.



Bitcoin: A Peer-to-Peer Electronic Cash System

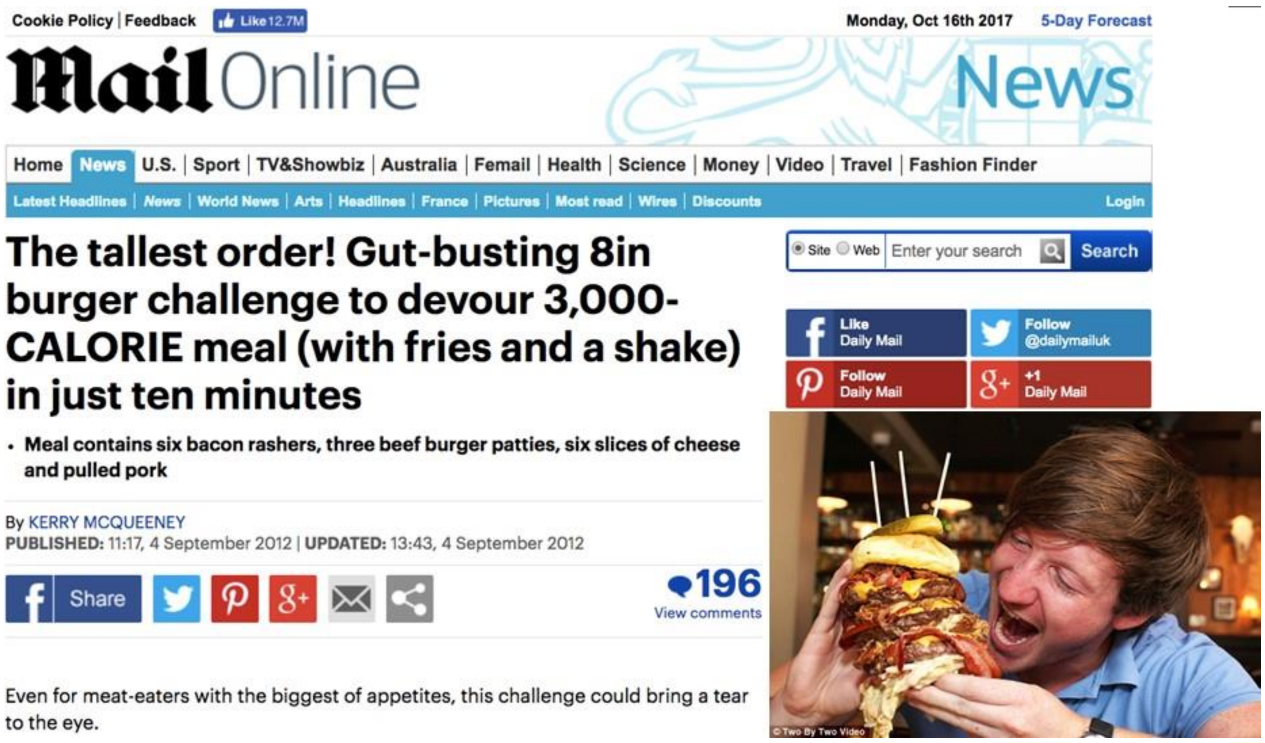
Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

Unfortunately, all of this thinking was quite happily ignored by the people who created actual today's smart contracts. [laughter] I have to credit this stuff, because this is the first widespread deployment of computer contracts in the real world, and of course it went ahead ignoring all of academia.

When Ethereum came along, it was fantastic: you could build unstoppable applications using things like Solidity... Who here actually knows Solidity? Okay, we've got a few hands. And Viper, anybody doing Viper yet? No, that's R&D. Then the DAO happened of course. Obviously this was a big embarrassment for Ethereum, and poor Vitalik had to make a choice. [laughter] It's funny, right? From a lawyer's point of view, this is why you need dispute resolution. Because this is a situation where if the only way to do a lawsuit is to have a full-blown constitutional crisis, then there is something wrong with your governance mechanism. [laughter]

There's actually papers on how to blow up the Ethereum smart contract, which is why we need better tools and need formal verification of Ethereum, and somebody is actually working on this today. Ethereum is not the only game in town, there are other languages that these guys... How much did they raise, $250 million? Partly on the strength of the fact that they had a formally verifiable smart contract language. Our friends in the room, Stephen has got his language, these guys are based in New York, they've got their language, and we've got the little thing that we've been working as well.


Cookie Policy | Feedback
Like 12.7M
Monday, Oct 16th 2017
5-Day Forecast
Mail Online
News
Home | News | U.S. | Sport | TV&Showbiz | Australia | Femail | Health | Science | Money | Video | Travel | Fashion Finder
Latest Headlines | News | World News | Arts | Headlines | France | Pictures | Most read | Wires | Discounts
Login
Site Web Enter your search Search
Like Daily Mail
Follow @dailymailuk
Follow Daily Mail
+1 Daily Mail
Share
196
View comments
© Two By Two Video

The tallest order! Gut-busting 8in burger challenge to devour 3,000-CALORIE meal (with fries and a shake) in just ten minutes

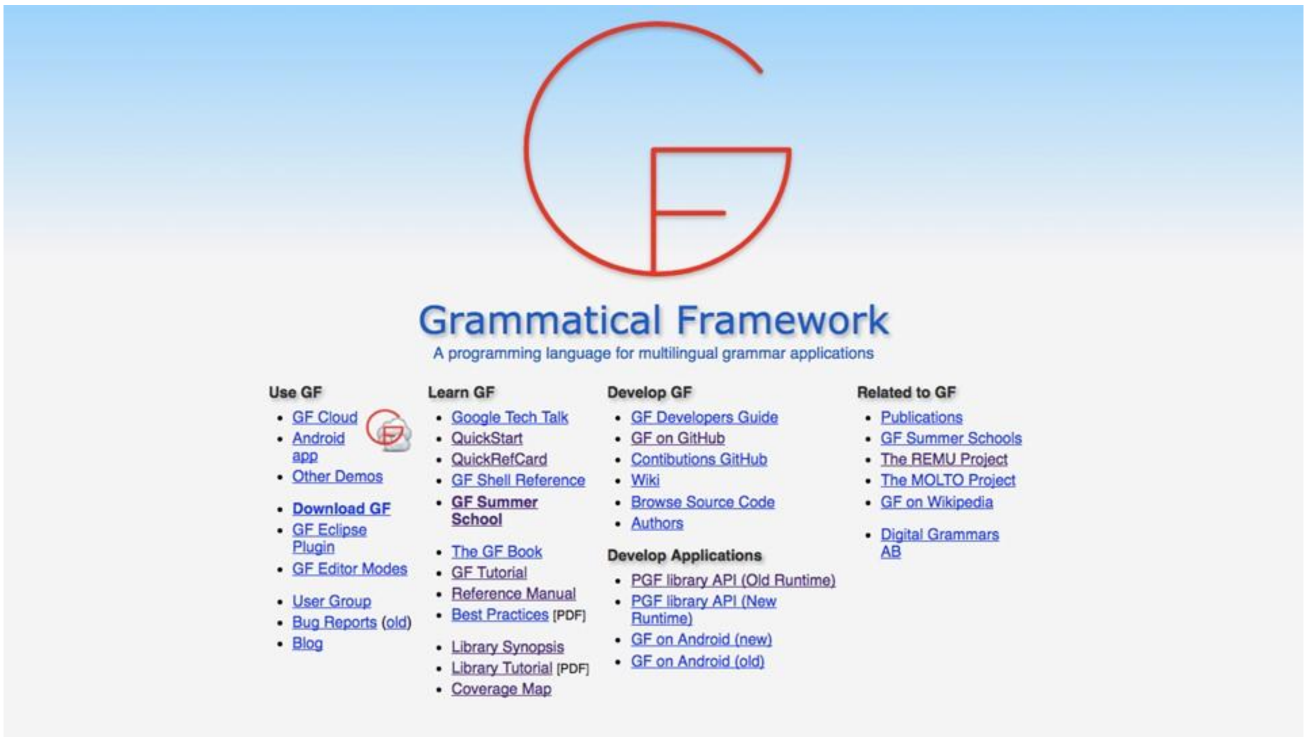
- Meal contains six bacon rashers, three beef burger patties, six slices of cheese and pulled pork

By KERRY MCQUEENEY
PUBLISHED: 11:17, 4 September 2012 | UPDATED: 13:43, 4 September 2012

Even for meat-eaters with the biggest of appetites, this challenge could bring a tear to the eye.

I'll show you an example of how we've been trying to formalise with our language, to show that it's not really a smart contract. We're not putting this on the blockchain, but this was the simplest "hello world" that we could imagine: it's a contract where if you can finish the burger, then your meal is free. [laughter] They've got one of these in London, so we can all do this after the conference.

But we're taking this seriously. As a use case, we are writing code to represent that burger situation, and the cool thing about this is that you can take this, you can compile it to English, you can compile it to Ethereum if that's what you want, you can formally verify it so that we demonstrate and prove that the customer isn't required to eat an infinite number of burgers, you don't want that, and all of this is going to be open source, it's on GitHub and it's all there.



Grammatical Framework

A programming language for multilingual grammar applications

Use GF

- GF Cloud
- Android app
- Other Demos
- **Download GF**
- GF Eclipse Plugin
- GF Editor Modes
- User Group
- Bug Reports (old)
- Blog

Learn GF

- Google Tech Talk
- QuickStart
- QuickRefCard
- GF Shell Reference
- **GF Summer School**
- The GF Book
- GF Tutorial
- Reference Manual
- Best Practices [PDF]
- Library Synopsis
- Library Tutorial [PDF]
- Coverage Map

Develop GF

- GF Developers Guide
- GF on GitHub
- Contibutions GitHub
- Wiki
- Browse Source Code
- Authors

Develop Applications

- PGF library API (Old Runtime)
- PGF library API (New Runtime)
- GF on Android (new)
- GF on Android (old)

Related to GF

- Publications
- GF Summer Schools
- The REMU Project
- The MOLTO Project
- GF on Wikipedia
- Digital Grammars AB

Let me just spend one minute to talk about how we produced the English and the German and Chinese and the French. There is an open source tool called GF, Grammatical Framework, and this allows you to program multilingual grammar applications. You take your smart contract, you transform the AST [abstract syntax tree] into something that GF can read, and in GF we've been modelling exactly the kind of "you must, you must not, you may, you may not," and we've modelled out that clause, and we are able to produce very simple test cases of "Under these conditions, this party must do this thing, meeting these requirements." This is kind of where we are right now, and I think in a year we'll be much further along.

LEGALESE.COM *SOFTWARE IS EATING LAW* 2017-10-16 INTERNET OF AGREEMENTS

THESE ARE REALLY SMART PEOPLE, *said one lawyer who works with startups.*

They believe in WORLD-DOMINATION *of the engineering class;* EVERYTHING *can be reduced to an* ALGORITHM *and* LEGAL DOCUMENTS *are not going to be* SPARED.

TECHCRUNCH

Thank You

slides available at goo.gl/mQkyTj

LEGALESE.COM

@mengwong @legalese
collective@legalese.com

CREATIVE COMMONS BY-NC