Developing a Legal Specification Protocol: Technological Considerations and Requirements

LSP Working Group

Draft of April 3, 2018

A NOTE ON THIS DRAFT:

This draft is not a fully complete version of the White Paper. Some portions are simply outlined and all of it is a work in progress. It is circulated in advance of a working day at Stanford's CodeX Center to provide a summary of the work in progress on the Legal Specification Protocol (LSP) project and to provide a starting point for our discussions. We anticipate that these discussions, and the suggestions and criticisms embedded in them, will provide the basis for completing the White Paper. In addition, this draft does not yet contain the full notes, references and bibliography that will be a necessary part the final product. Those additions will be made in future drafts. For further consideration of the process and next steps within it, we refer you to Sections I.B. and V.C of this draft.

I. Introduction: Background and Need for a Legal Specification Protocol

A. Time for a Legal Specification Protocol (LSP)

In many domains of human activity, the application of electronic computing has made once cumbersome tasks quicker and easier. The automation of portions of legal and regulatory processes holds the similar promise of delivering faster and better service at lower cost. Existing public and private initiatives on legal technology ("LegalTech") have made some progress in this, but they have been held back from delivering the full possible benefit by the lack of a widely shared protocol for expressing legal formulations in executable code.

The development of the Internet gives a useful comparison. In the early stages of linked digital interaction, there were a number of competing, siloed initiatives, Compuserve, Prodigy and AOL in the US and Mintel in France perhaps most prominent among them. These initiatives gave some service, but the full flowering of the Internet as a platform required establishing the Internet Protocol Suite, an architecture model and a set of shared conventions for representing and transmitting data that could be utilized and built upon by a number of different applications.

We are at a similar crossroads in legal automation, where a number of competing, and often privately established, approaches exist for expressing and processing legal and regulatory formulas. Law, at its best, however, is grounded on shared, open access conventions. By way of example, in the United States "Blue Book Form" creates a shared standard for classifying print-based citation data. To succeed fully, legal automation requires a similar shared foundation. Legal automation needs to get out of the AOL phase and on to the World Wide Web. The time is right to make a concerted effort to develop an expressive Legal Specification Protocol (LSP), that will` have the capacity (i) to capture the event space salient to legal formulations, (ii) to represent the computational and logical structure of legal specification and (iii) to allow the execution of the process and workflow imbedded in that structure to provide useful applications to a broad swath of legal tasks, including contracting, compliance, and legal judgments.

In addition to the ability to accomplish these tasks, this protocol should enable relatively easy use by both legal professionals and the general public through accessible and well-designed user interfaces. It should also be designed, to the extent possible, with the ability to interact with legacy systems and with the ability to grow to represent new formulations and to meet new needs. Interoperability and generativity are as important as expressivity for the protocol.

Because of the public goods and coordination challenges inherent in this kind of general development project, the marketplace has been slow to create such an LSP. This is therefore a good target for development through a collaborative process involving private sector actors, academic centers, philanthropy and governmental standard setters.

As will be discussed more fully in Section IV below, the LSP project can rely on steps already taken in projects such as OASIS' Legal XML, CALI's A2J Author, CodeX's CompLaw initiative, and other public and private initiatives. We do not want to reinvent what can be usefully incorporated. That said, these initiatives have typically solved only portions of the problem, and some haven't been designed with the full expressivity of broad-based legal specification in mind.

B. Project Stages

As envisioned by our working group, the project should be developed in stages. The key steps for this initiative involve:

1) spec'ing out the needs of legal expression and of a LSP that would enable computational machines to represent and execute that expression;

2) surveying existing projects and approaches;

3) creating and convening a network of collaborators; obtaining their input on 1 and 2, establishing use cases for focusing our efforts; and seeking contributions of labor and of financial backing for the next steps;

4) building on 1, 2 and 3, creating of the Legal Specification Protocol and demonstrating its application to selected use cases;

5) creating the user interface;

6)creating a platform on which 4 and 5 can operate; and

7) the disseminating the language and platform to commercial, academic, governmental, and NGO users.

These steps cluster into three groups – stages 1-3 make an initial phase, 4-6 make a second phase, and 7 makes a third and final phase. This white paper is intended to provide a summary of work to date on the first phase, and provide a bridge to beginning the work on the second.

In addition to its technological capabilities, the LSP needs to meet social criteria to prove effective. To begin with, the LSP should be a public utility, not owned or controlled by any particular private or governmental entity. There is a long history of law operating best as an open-source system, where citizens, businesses, and governments can all access its rules and use its processes. That said, there should be ample room for private enterprise to make use of the LSP to create applications with commercial potential.

Once again, the Internet provides a useful example. The basic architecture of the Internet is open to all for use; applications from Google to Facebook create proprietary layers on top of that architecture. This is an appropriate division of the differing public and private goods structures of protocols and applications. Because of the public goods nature of the LSP itself, it cannot be expected to emerge from private, competitive businesses left only to their uncoordinated interest.

As will be discussed more fully in Section V below, we are pursuing the LSP not through some fully structured, hierarchical initiative, but rather through a coordinated but also distributed and decentralized set of efforts. In this we are seeking a somewhat unusual pathway that we feel will be productive for the particular needs of this project.

C. Contracts as a Use Case

In these early phases of activity, the LSP working group has chosen the representation of Contracts as our focusing use case. There are a number of reasons for this choice. To begin with, contracts present a relatively tractable problem. Contracts are typically shorter, more discrete legal formulations than legislation or regulation, while still containing, in these shorter formulations, most of the logical complexity present in other forms of legal specification. Because contracts typically involve only a few parties, often private actors, the coordination problem of getting acceptance of a contract protocol is much more tractable than it would be for an entire legal system. Also, contracts often are associated with relatively high value transactions, and reducing the drag of classical natural language contract formation, interpretation, and implementation can provide savings that will help incentivize adoption.

For all these reasons, computational law “privileges” contracts as a starting point – consistent with the notion that contracts are private law, “privi leges”.

Within the field of contracts, the working group anticipates selecting even more specific domains of contracting as use cases for informing and testing the process of developing an LSP. As will be explored more fully in Section II.B of this paper, we are currently targeting several specific contract categories for this purpose.

D. This Paper

This paper and its accompanying appendix are intended to sum up the conclusions of the working group on the first two stages set out above:

1) spec’ing out the needs of legal expression and of a LSP that would enable computational machines to represent and execute that expression; and

2) surveying existing projects and approaches.

In defining the requirements, this paper will first explore the uses and functionalities of traditional word-based contracting. Delving deeper, it will overlay that analysis on several specific use cases. Next, it will look at the computational tools and approaches that could be applied to embody these needs. After a summary of computational approaches generally, it will divide the description (somewhat artificially) among 1) the logical expressivity needed to represent contracts; 2) the informational expressivity needed to allow the processing of contracts and their interaction with the world they look to regulate; and 3) necessary components of a user interface.

This paper will then provide, in Appendices A and B two surveys of existing efforts on contract and legal automation, and in the main text will focus on several efforts that we believe have particular promise for better informing our LSP approach. Finally, we set out a summary of the LSP activities to date and outline a plan of action for going forward.

II. Focus on Contracts and What They Do

A. The Tasks and Parts of Contracting

1. The Functionality of Contracting

In order to be useful in creating contracts, a legal specification protocol will need to be able to represent the things that contracts do. While there are many approaches to theorizing contracts, from a functional standpoint contracts can be seen to provide an actionable outline for committed performance in a multi-party relationship. The specified actions can be affirmative: things which a party is promising to do. The specification can also create boundaries: things which a party will not do, such as a non-disclosure or limits on the exercise of rights granted.

Contracts also provide for alternative pathways if there is a breakdown in the expected outline of action and restraint. For instance, if there is a breach in a financial contract, a new pathway kicks in, providing for notice, possible cure, acceleration of obligations, rights to take over collateral, and other new and different actions and limits. The tripwires that set off these new pathways can be either directly linked to the expected performance or can be linked to important ancillary facts, events, or actions, through mechanisms often called “representations and warranties” and “affirmative and negative covenants.”

Finally, contracts can provide guides for enforcement and consequence if there is a sufficiently serious breakdown in the expected outline. Choice of law, forum, and venue fall into this category, as do specifications about remedies, damages, and indemnification.

Why do we specify such chains of action, breach and consequence? The commitment structure of contracts helps to solve many of the strategic dilemmas that stand in the way of productive cooperation among parties with potentially divergent interests. The "first mover" problem is one such dilemma. In this structure, potentially costly actions need to be taken sequentially by the separate parties. The sequence leaves the first mover vulnerable, as the second mover may fail to take the expected reciprocal step, or, even worse, may actively appropriate the benefit of the initial move in step of predatory exploitation. A contract that spells out the expected steps of both parties and links a failure to take a step to costly consequences can help keep both players honest and make the innocent party financially whole if there is a lapse. Such private "institutions" are the foundations of much of the cooperation that makes human progress possible.

2. The Techniques of Contracting

The classic techniques of contracting are tied to the functions and challenges described above. In most instances, natural language statements provide the formalism for setting out the combination of events and outcomes that constitute the commitment structure of the contract. The starting point for most contracts is a statement that sets out the expected path of execution for the parties. In the simplest of contracts, this is all the parties set out, leaving questions of breach and enforcement to the framing of general legal principles. In more complex contract practice, this statement of the expected path may be set out largely in an addendum prepared by the business people who will be responsible for carrying out the promised actions, while the contract proper, under the care of the lawyers, sets out the triggers and alternative obligations applicable to a contractual breach. These alternative pathways correspond, to a significant degree, with the concept of contract exceptions discussed by Grosof and Poon (2004).

Much of this "happy path" of expected action can be represented in the business process management software of the firm, whether aimed at manufacturing and delivery or at financial calculations and payment, and these steps have already been a significant target for automaton. Ordering through the Amazon website is an example of this put into practice, as are most so-called "smart contracts" in the cryptocurrency world.

Some contracts specify not so much actions as boundaries or limits on actions. For instance, a distribution contract for a film or other audiovisual production may set out limits for the rights granted, set to territories, media or time limits. Here, rather than requiring certain actions, the goal is to forbid them. Do not distribute outside the United States, or through cable television, etc. If one of these tripwires of constraint is activated, then, as before, we enter a default cascade that could include cancelation, damages, and other sets of remedies.

More complex contracts can spend a great deal of attention on the definition of the circumstances that will invoke an alternative pathway. Such a switching event is often termed a "default." In classic contracting, a default can result from triggering events in three different contexts. The first, and most clear-cut, is some kind of significant failure in the direct performance specified in the "happy path." Often there is an intermediate step, aimed at putting the performance back on track, where notice of the problem is provided together with an opportunity for "cure" – actions to bring the contract back into its expected pathway. If cure isn't affected, or if the breach is severe enough that cure is pointless, then the contract may move to a new set of specified events and actions for a state of default.

A failure of the principal obligations is not the only circumstances causing breach and default. In addition, a complex contract can also specify a set of ancillary performance criteria, often called "covenants," as well as a set if ancillary facts and states of the world, often called "representations and warranties," that will serve as switch-points. For instance, in a financial contract such as a loan, the borrower may promise to stay current on all tax payments. This covenant is not directly related to the making of payments back to the lender, which may still be flowing smoothly. It does, however, potentially change the risk profile of the transaction, which could, in turn, make the lender want to accelerate the payback period to make the whole loan immediately due. Likewise, a change in the tax law, not related to any action by the parties, could shift the nature of the bargain and trigger the applicability of a new track.

A great deal of lawyer time in complex contracts is devoted to arguing about the contents of covenants and representations and warranties together with the string of consequences when there is a breach of one of these ancillary boundaries. These lead naturally to further negotiation over how the obligations of the parties, whether under the expected deal or under a different set of default induced terms, will be enforced. Sometimes various kinds of automatic consequences can be built in, like the removal of access to a website or the release of an escrow fund. In other cases, societal intervention through an enforcement proceeding may be necessary, with the invocation of some form of dispute resolution forum and the eventual empowerment of the Sheriff or Marshal to seize money or even make an arrest as the final consequential back-stop.

3. Lawyers Constructing Contracts

So how do lawyers construct contractual instruments that embody this functionality? Currently, they most frequently start with existing templates or sometimes design new word-based formulas for constructs of event and outcome. The drafting uses natural language to specify events or event strings linked by simple connectors and then string them together into if/then or then/else conditional statements that chart the expected "happy" pathway as well as the default linked alternatives.

Lawyers often attack such drafting problems intuitively, without expressly understanding the process they are undertaking.[1] One of the side benefits of examining the computational structure of contracts is that it helps a practitioner to clarify what is going on, even without taking the step of actually encoding these statements into software. And one of the requirements for a useful contract specification protocol will be a user interface that will allow the easy construction of structures such as these.

4. The Ambiguity Challenge: A bug or a feature?

In discussions of legal automation it is often pointed out that word-based specification of contracts and other legal formulations can contain "ambiguity," that is not-fully-defined aspects that leave matters open to some interpretation. Lawyers sometimes view this as a desirable design element and suggest that a barrier to contract automation is that by using software, much of this ambiguity will be clarified. This is a mistaken assertion on a number of levels. To begin with, it is necessary to understand what is meant by ambiguity in a contracting context. A useful distinction can be made between "rule ambiguity" and "event ambiguity." Rule ambiguity involves a circumstance where the pertinent facts are well established but where the outcome resulting from those facts is not clearly specified. Most drafters – and their clients – would not view rule ambiguity as a plus. The point of contracting is to project dependable order into a set of future interactions.

Event ambiguity is more pervasive: has the trigger happened? If so, then the consequences flow. But determining events is notoriously difficult. In computing more generally it is sometimes termed the "feature extraction" challenge. Think facial recognition software – is the person viewed really Kim Kardashian or someone else? If a Kardashian, implement plan A, if not, plan B. Trick is the recognition, not the reaction. So it goes for a great deal of contracting, and the law generally, for that matter. Careful definition of the salient events, such as delivery, insolvency, or LIBOR, together with the means for determining their occurrence, are among the more important tasks of creating good contracts. The intentional injection of increased ambiguity into this process can occur, but it runs counter to the purpose of contracting.

A further characteristic sometimes called "ambiguity" is the intentional use of events that require human judgment to resolve. A contracting party will use "reasonable efforts" or will

[1] [1] Precision's Counterfeit: The Failures of Complex Documents, and Some Suggested Remedies, Howard Darmstadter, The Business Lawyer, Vol. 66, No. 1 (November 2010), pp. 61–83

make “timely delivery”. Such time-honored shorthand spares the parties the task of breaking the criteria down into ever more granular elements of measurement, substituting, instead, reference to human judgment as the decider. This is not so much ambiguity as it is invoking a particular human “black box” as the event trigger. Once the authoritative human or humans (e.g. judge or jury) determine that reasonable efforts were used, then the application of the rule and the determined outcome becomes clear once again. The instruction “ask the designated human” is a perfectly acceptable (and non-ambiguous) step in a well-constructed contractual logic tree, whether specified through words or computer code. Such a designated human is, in law, sometimes a judge but sometimes an “expert witness”; in blockchain parlance, the human or committee of humans is an “oracle”. AI judges are NOT a necessary component in a computationally assisted and specified legal system.[2]

5. Links to Legal Specification More Generally

While the emphasis of the LSP initiative, and this White Paper, has so far been directed primarily at contracting, we believe that an approach developed to capably enable contract specification and execution can serve as the basis for an approach that will work for legal specification more broadly. This is particularly true for the explicit rule formulations generally found in legislation and regulation, and which particularly characterizes the civil law system. Common law rule making, at least within the Anglo-American tradition of looking at reported court cases for guidance on future outcomes, may require additional steps for representation. The two traditions reflect, to some extent, the difference between rule-based formulation, building chains of event and consequence intentionally from the ground up, and data-based rule extraction, where the pattern is recognized in the data and then formalized into some kind of rule. Case based reasoning is largely in this latter category, although over time its outcomes become more and more rule like, as when the “restatement” process produces abstracted principles that represent the holdings, and which get cited as authority. In this context, our efforts may be broadly conceived as a Computational Restatement of the Law of Contracts.

B. Use Cases

1. Use Case Goals and Stages

The development of a widely used Legal Specification Protocol cannot take place as an abstracted exercise. The protocol should reflect the actual needs of contracting and other examples of legal specification, and that will only occur if the development of the formalism is

[2] Ambiguity in contracting has been discussed elsewhere, e.g. by Claire Hill in *Bargaining In The Shadow Of The Lawsuit: A Social Norms Theory Of Incomplete Contracts* and in the work of Oliver Hart and Bengt Holmström for which the Nobel Prize in Economics 2016 was awarded.

securely anchored in particular cases. The goals and stages of working with the use cases reflect:

A *first stage* involves breaking out the expressivity needs for both the information specification and the logic of the processing architecture. In this stage we will take particular contracts and break them apart, cataloging the information which they consume and emit and the structure of the logic that links specific inputs to designated outcomes. For instance, in their paper on contracts as a computational statement, Flood and Goodenough[3] isolate 20 distinct events, ranging from time passing to going bankrupt, as constituting the information which a short financial contract will potentially consume in its life cycle, including a limited set of default triggers and consequences.

We will take the targeted use case contracts and perform a similar listing of salient events. We will then go the next step with each of them and seek to break those events apart further into their components of time, measurement, performance, and conclusions from other reasoning process. The goal in this is to identify the minimum necessary taxonomy of content types that will capture the information requirements of a legal computation. This taxonomy will include the output of such a computation as well as the inputs. An initial formulation for such a taxonomy is set out in Section III.B of this white paper.

On the processing side, we need to identify what logical connectors are necessary to express legal/contractual reasoning, as well as connectors that can powerfully bundle together the concepts. For instance, it may be possible to use if/then/other style reasoning to express most legal/contractual logic steps, but there may also be “shortcuts” from more advanced logics, such as deontic logic and defeasible reasoning, that will shrink a process needing many steps in if/then/other to relatively few steps – or even just one – in a more powerful logic system. The architecture of shifting from the anticipated payment structure to an acceleration on default in a financial contract could be a target for such a step. The logic must also match up with the information elements described above.

The end product of this exercise, played out against a small but diverse set of use cases, will be a scoping of needs of the protocol, an necessary precursor to beginning to build it as a useable approach to legal computation. This process will also be informed by existing approaches to specifying legal reasoning in computable terms.

A *second stage* will be the creation of the actual protocol and the building of software representations, based on the protocol, of the contractual use cases. Here the project will move to direct application. The process is likely to be iterative, along the lines of the Agile

[3] M. Flood, and O. Goodenough, “Contract as Automaton: The Computational Representation of Financial Agreements,” OFR Working Paper, No. 15-04, March 2015.

model, with a movement back and forth between the creation, application, testing and feedback elements. Indeed, the distinction between the first and second stages set out here is partially artificial, as the early parts of this second building stage will overlap with, and provide feedback for, the scoping process.

2. Identifying Use Cases

At this stage we have targeted several use cases and are looking to add one or two more to our list. The criteria for selection include:

- Domain knowledge in our working group
- Availability of partners who are active in the target area, and other resources for the project
- Potential for early gains from increased computational representation in the target area
- Likelihood of buy-in by players in the target area when the use cases come to fruition
- Some diversity of subject matter, so as to test the protocol against a variety of legal domains
- Outcome importance, which can be judged by several criteria, including social utility, promotion of justice, financial importance, and the potential for impact.[4]

Applying these factors, the initial targets are financial contracting, service level agreements, early stage investment [manufacturing supply chains, and international trade with a link to blockchain].

2.1. Financial Contracts Use Case

Context and objectives

The valuation and risk analysis of individual securities, or of a portfolio comprising multiple securities, is a fundamental task. This basic task can operate at many levels in the financial system. For example, contractual exposures exist in the portfolio managed by the securities lending desk at a dealer firm, the combined portfolio of all available-for-sale securities across all the trading desks in the same firm, the full balance sheet of that firm, the network of positions among that firm and all of its competing dealers, or the financial system as a whole.

[4] These factors helped to inform the use cases working group at 9/9/17 working meeting that led to identification of salient candidates. For more, see notes from 9/9/17 Use Case Working Group compiled by Tony Lai: https://goo.gl/xzbKxr

An important common element to the comprehension of the problem at these various levels is the analysis of individual financial contracts. If we can represent diverse contracts with a parsimonious common language, then it should become possible to develop tooling, such as specialized “contract databases” or risk analytics that work not only for a fixed set of known security types, but equally for arbitrary financial contracts, some of which may be innovations yet to be invented.

Prior art in this domain includes lexifi.com, Hyperledger Cicero, http://www.actusfrf.org, R3 Corda, and recent work in translating ISDA master agreements for execution within a DLT or blockchain.

In particular, by exploiting a set of basic abstractions with broad applicability – namely the LSP – it should be possible to develop a map of the financial system, or of an arbitrary component or subsystem. When fully populated with relevant data, such a map would enable a user to navigate through arbitrary chains of legal and financial risk exposure, to group arbitrary subsets of contractual positions into meaningful aggregates, to drill down into the fine-grained details of individual contracts, and to find important patterns and concentrations in contingent exposures across an arbitrary subsystem under consideration.[5]

Users and stakeholders

The primary user in this scenario is a financial analyst tasked with examining some set of financial contracts – for example, all corporate bonds held by pension funds supported by the Pension Benefit Guarantee Corporation (PBGC), or all credit default swaps held on the books of systemically important financial institutions (SIFIs) – to identify any specific events that would trigger cash flow obligations that, in the aggregate, exceed a threshold level of $50 billion. Moreover, the analyst should be able to whether and how such obligations might be linked in the network (e.g., A owes B owes C, etc.) in ways that could create contagion.

Usage scenario

- The analyst launches a contracts database (with related query tools) that enables her to load detailed contract data from diverse sources into a common repository that uses an LSP schema.

- The analyst logs into a source reporting system that provides contract data in the raw format as provided by a set of pension funds.

- The analyst applies an extract-transform-load (ETL) script to the source data to normalize it from its raw format to the LSP schema. This executes as set of parallelized

[5] For bibliography – search legalese.com's "70 years of R&D" for the term "financ"

batch jobs running across the raw source data stored in the reporting system. The output of these processes is stored in the contracts database.

- She launches a query/mapping tool and connects to the contracts database.
- Using the LSP contracts query language (CQL), she selects the desired set of pension fund contracts. She then performs a second query on this subset to identify any contingency clauses that result in cash flows, as well as the event definitions that could trigger those clauses. By sorting and aggregating, she is able to identify a small set of events that would trigger cash flows above the predefined $50B threshold.
- She pulls the associated contracts from the database and loads them into a visualization tool, which highlights the obligations by event type. For each event type, she runs contagion analytics to generate a contagion probability scored, and she generates a summary image that illustrates how those event-dependent obligations are linked in the network.
- She produces a short report, including the contagion scores and network images, for

2.2 Telecom Service Level Agreements

TM Forum (TMF) is **an industry association of 850 member companies worldwide of communication and digital service providers and their ecosystem of supplier**s that focuses on digital transformation via various collaborative efforts. TMF has an interest in exploring computational law, particularly in the context of its exploration of **blockchain**. TMF developed a proof of concept incorporating both, featuring an SLA, which it demonstrated at its 2017 annual conference in Nice, France, attracting interest from member companies which have since moved it forward as one of the use cases addressed by the "**Blockchain Unleashed**" project within the **TMF Catalyst program**. Meanwhile, at the September 9, 2017 LSP convening Use Case Working Meeting, the telco SLA use case was considered a viable candidate given its potential value and utility, as well as potential industry support to build and adopt based on TMF interest. The description of the use case below is based mostly on the slide deck, "Blockchain Sandbox: SLA Management" (filename TM Forum Blockchain Sandbox 2017-09-22.pptx), developed by John Wilmes, TMF director of IoE projects.

Context/Users

From Slide 12 of the above-referenced deck:

- Service level agreements (SLAs) and related business processes have been exhaustively defined by the TM Forum and other organizations, but still incur risk of litigation over significant sums.

- Sources of risk include
    - Legal agreements that are discovered to be inconsistent, incomplete and/or incorrect
    - Processes that do not reflect legal agreements
    - Disagreements over the application of service level criteria and/or measurements
    - Long intervals between settlements leading to “bill shock”.
- The demo described here attempts to demonstrate the technical possibility of addressing the above sources of risk:
    - Use computational law tools to ensure the validity of the legal contract and its mapping to the smart contract - not demonstrated as of July 2017
    - Use a blockchain to provide a shared, immutable record of criteria, measurements and events as well as a platform for smart contracts - demonstrated
    - Use smart contracts running on a blockchain to automate the detection, assessment, documentation, and settlement of SLA violations – demonstrated
    - Use token currency supported by a blockchain to make frequent, small settlements using token micropayments – demonstrated

From TMF **“Blockchain unleashed” user story** (4/1/18):

As a provider of digital services to enterprises

I need to ensure that my service level agreements are created, managed and executed in a way that minimizes friction and risk

so that I can rely on timely settlement and absence of litigation.

To do this I need to ensure that my service level agreements are complete, consistent and correct and that their execution and settlement can be automated.

I know that I am successful when my service level agreements are automatically executed as smart contracts.

Usage Scenario

Slide 21 of the above-referenced deck describes the usage scenario; it is currently anticipated that the LSP initiative may be most relevant at numbers 1 and possibly 2 below.

1. Legal staff use a template to create a contract representing a service level agreement (SLA), validate the contract using a computational law tool, and distribute it for electronic signature.
2. Technical staff use the validated electronic contract and a smart contract template to create a smart contract that will manage the execution of the previously defined SLA.
3. The SLA covers the availability of a Web site as reported by a smart oracle. Pingdom, a widely trusted company for this purpose, monitors the Web site. The smart oracle uses the Pingdom API as a trusted source of data.
4. The two parties to the SLA each place in escrow an amount of ether sufficient for worst-case fulfillment of their contractual obligation. They then initiate the smart oracle and smart contract.
5. The smart oracle reports the smart contract initiation to the SLA management system using the TM Forum SLA Management API.
6. The smart oracle sends the smart contract a message at previously agreed intervals. The message contains the current up/down state of the Web site.
7. The smart contract makes incremental micropayments from the escrow to one of the parties at each predefined interval, based on the reported state of the Web site.
8. The smart oracle reports violations to the SLA management system using the TM Forum SLA Management API.

As of 4/2/18, the "Blockchain Unleashed" project provides a usage scenario description in its video from January 2018 posted at https://projects.tmforum.org/wiki/display/CS/Blockchain+Unleashed.

2.3 Early Stage Investment

Another use case in the financial domain is the automation of early stage startup investment agreements. Securities such as convertible notes, SAFEs, and KISSes are just complicated enough to be interesting: they are simple enough to be captured in a relatively straightforward structure describing the parties, monies, valuation cap, and conversion discount; yet they interact with subsequent events (i.e. a Series A equity financing, which may itself be represented computationally) in a complicated enough way to derive benefit from automation.

In more detail: a technology entrepreneur contemplating a fundraising from a group of angel investors may balk at the legal fees involved in retaining a traditional law firm. She may be aware that free templates are offered by Y Combinator (the "SAFE") and by 500 Startups (the "KISS"). As a technologist she may prefer a fully computational approach: rather than downloading the Word templates and manually instantiating one document per investor, she may prefer to describe the entire fundraising in a spreadsheet or even a JSON object which obeys an LSP schema. Once fed to a document automation backend, such a structured LSP data object should automatically call forth all the needed paperwork for the angel round – and all the needed paperwork for conversion at the future Series A round. Beyond document automation, such an LSP-compatible approach could offer the entrepreneur the option of better understanding the deal terms and consequences through software-supported scenario visualization and ontology education, which would ordinarily be professional services provided at a high hourly cost.

[2.4 Supply Chain Management]

[Can be linked to Industry 4.0 initiatives. Can build on/collaborate with IACCM and Accord Project in this area.]

[2.5 International trade with a link to blockchain]

[Building on blockchain bill of lading approaches]

[2.6 Additional possibilities to be developed at the April 6, 2018 meeting]

[In addition to these use cases, we would welcome the identification of one or two others, meeting the criteria set out above and in particular providing some further diversity of subject matters. We believe that the Industry 4.0 initiative could provide just such a target, capturing the needs of contracting in supply chain, manufacturing outsourcing, and other similar contexts and providing a basis for further and more effective automation of these processes.]

III. Describing the Computational Requirements/Tools for Accomplishing these Tasks

<u>[A. Logic and programming of legal event and consequence Outlined Here; Full ext in Development]</u>

Over the years, many logics have been suggested for legal representation.

The list includes Deontic Logic, specially adapted to moral statements with components such as "ought" and "should". There is some disagreement among the working group on the utility of deontic approaches to contracting.

Some argue that current word-based law does not go there; in fact, statutory, regulatory and contract prose AVOIDS language of moral obligation and instead uses direct statements of event and consequence

- Rather, this prose sets out a string of events and consequences that reset the game structure to make desirable behavior more likely.
- The Holmes "Bad Man" theory of law. "Bad men, Holmes argued in his speech "The Path of the Law" care little for ethics or lofty conceptions of natural law; instead they care simply about staying out of jail and avoiding the payment of damages. In Holmes's mind, therefore, it was most useful to define "the law" as a prediction of what will bring punishment or other consequences from a court." Wikipedia
- Others argue that which is the same approach that deontic logic takes: if there is no consequence to the violation of an obligation, then the obligation does not exist in any material sense.
- It is worth noting that in deontic logic, every permission granted to a party may be equivalently stated as an obligation upon a counterparty to respond to the voluntary, permitted action by that first party.

For the most part, current legal drafting, including contracts, breaks down into relatively simple conditional logic, such as if/then or if/then/else statements, with the highest level of complexity being terms like "if, but only if", creating a natural language "pseudo code" we call legalese.

- When very complex it is often that the string of events/facts can be quite long, and full of and/or and other connectors stringing them together
- The various provisions can form "chained conditionals".
- Also can be broken down into nested sub-routines/objects
    - A defined term, used in several places, serves as such a computational object
    - As can a cross-reference
    - As with object oriented programming, cuts down on recursions
    - Also nest up against other layers with implied application, such as the rules of procedure should enforcement by required
- See, e.g., the material at https://curriculum.code.org/csp-1718/unit5/9/#if-else-if-and-conditional-logic1
- So in the law we see complex statements (e.g. the Tax Code) built out of extensive terms applying relatively simple logic in highly involved ways

    - Defeasible logic – applicable to the multi-path structure, and potentially useful in that creation, but is it *necessary* as a separate classification for contract programming or can it be captured by conditional logic?
        - Quite useful for designing AI; perhaps less necessary, other than as a shortcut, for designing the underlying logic of an LSP
        - May help the design at the UI level[6]

A lucky aspect of the law's use of conditional logic is that conditional statements of this kind can be represented in a wide range of existing computer languages and approaches.

- We probably don't need a specific new legal representation language per se
    - Special approaches may eventually be useful for creating wider ranges of legal specification
- We do need to develop versions of languages that can efficiently represent the specific kinds of tasks that law generally and contracts specifically require; see the discussions above.
- And we need versions that can interact with the shared data and communication standards that will permit interoperability and with the input and output UI layers that WILL need to be specifically law oriented.

B. Data/Information needs

Law, as a computational system, consists of a set of rules about specific "events" which change the current state of the system through a set of logical transition steps to produce conclusions about consequences. Even in the "slow AI" of traditional natural language-based systems of law, legal conclusions under contracts, case law, statutes and regulations follow this pattern of establishing relevant facts, seeing how those facts trigger legal analysis, and then reaching a conclusion of action, obligation, etc.

In order to enable greater automation in legal specification and analysis, it will be critical to create a widely used convention for describing events that are consumed and produced by computational law. This need exists both in the specific use case of contracts as well as in the broader case of legal specification generally. Such a standard is a necessary element in the "stack" of any widely-shared and sufficiently expressive legal specification protocol.

Set out below is a summary description of a set of elements that meets the informational requirements of competent legal automation approaches. The adoption of conventions for

[6] The UML tradition of business logic has produced such things as BPMN, BPEL, SBVR, and DMN which are diagrammatic, flowchart-like expressions of such rules, intended for business users to grasp more visually than textually. See Section IV.B.5 below.

such a system should enable a variety of software systems to digest legally salient events, to process them to outcomes in light of the logic of the legal rules involved, and then to produce statements of outcome that can be consumed and acted on, in turn, by other human and machine processes, such as workflow or payment systems.

1. Element List

The following list sets out an attempt to distil down a relatively comprehensive taxonomy of the information requirements that law sets out as salient as the predicates and outputs its if/then/else computational processes. This taxonomy is aligned with prior work by the Estrella project expressed in LKIF, the Legal Knowledge Interchange Format, with the OASIS work on LegalRuleML, and with the ontological requirements described in *Approaches to Legal Ontologies and in Legal Ontology Engineering*, although its categories have been developed independently.

- i. A statement of the event/data type
    - a. Can reference dictionary/taxonomy/etc.
    - b. Reference can be to a natural language description, but need not be
- ii. Value information about the event/data type
    - a. Can be yes/no, a measurement, a conclusion, a location, etc.
    - b. Can also include value related data, such as confidence level
- iii. Provenance/Source
    - a. Can be a particular sensor, a blockchain record, a court determination, the product of a particular prior computation, etc.
- iv. Time/Date stamp
    - a. State in universal time
    - b. Relate back to provenance (could be a sub-field of iii)
    - c. Distinguish event time and report time
- v. Matter
    - a. Particular contract, court case, application, legal citation, etc.

vi. Specification of the event in other systems (aimed at creating interoperability making it legacy friendly and a bridge between existing and new platforms)

a. A designation of the other system(s)

b. The designation, value, etc. coding within that system

vii. Security element (hash, certificate, etc.)

viii. Other

a. Open fields for things not currently imagined – subject matter extensibility, such as monetary currency, classes of company shares, and the particulars of financial instruments, to the extent these do not fit the taxonomy above.

2. Format for Specifying the Elements Can Come Later.

We are not, at this time, suggesting a particular architecture for specifying these elements. Some kind of articulated character string seems a likely candidate. Other domain-linked specifications, such as html or XML, can provide starting points for working out the architecture once the necessary/desirable elements are agreed upon.

C. Machine and Human Interfaces for Contract Creation and for Linking to External Input and Output

While no special logic may be necessary for a great deal of legal/contractual specification, presenting the logic for easy, relatively intuitive use and construction by contract creators will need to be carefully designed. In pursuing this, we need to collect more information on aspects of the design criteria. These include:

- How do people intuitively construct the chains of event and consequence that underlie contracting?
- How could we quickly educate people to use interface tools that would be quicker and more effective than simple intuition?
- How do we convince users that in a multiparty context that individual subjective intuitions are likely to conflict, and that objective interfaces accelerate the achievement of a “meeting of the minds”?

We may find that our design approaches are informed by more complex logical approaches, such as defeasibility, particularly if we find complimentary capabilities in human cognition.

In addition, we need to delineate in more detail the processes and particulate components of contractual relations at a step up from the simple conditional statements that can embody contracts. Take this common element as an example:

- What is an affirmative covenant?
- What are its common elements of event and consequence?
- How do we monitor the key event or state and bring that information into the contract?
- How does the covenant interact with the greater contract?
- What kind of visualization/dashboard would we want to keep track of the performance/nonperformance of the covenant?

With such an understanding, we can make the building blocks apparent and accessible within the UI, and represent how they fit together to create typical contractual statements. We anticipate that a usable legal specification protocol will at least utilize a symbolic, graphical interface, if not in fact a drag and drop interface similar to Scratch or to Decisions in the business process management world. Prior art includes BPMN / SBVR (OMG), Contract-Oriented Diagrams (Schneider), and work on visual contracts by Helena Haapio and Stefania Passera.

Because contractual elements relate to each other across a number of parameters, the interface may need to have the capacity to keep several elements or dimensions open simultaneously:

- What input of events or facts matter, including sources for the information
- Workflow style constructs of what follows after what in a particular chain
    - If/then etc.
- What outputs need to be kept track of, exported
- Major switch points and their triggers

The ultimate requirements will be a user-friendly design/programming/”document” creation space for the design tool, with the capacity to channel output for execution by both humans and machines.

IV. Existing Technological Efforts

A. General

There is a long history of work on the representation of legal statements in executable code. Flood and Goodenough provide the following summary:

> Our approach follows Allen (1957) and von der Lieth Gardner (1987), who similarly represent legal constructions with symbolic formalisms. Allen (1957) applies symbolic logic to legal documents generally. Although symbolic logic is more expressive and more general than automata, it is also a lower-level abstraction and inefficient for capturing the legal semantics of financial contracts. Von der Lieth Gardner (1987) considers computational representations of legal knowledge and reasoning generally. She considers an ambitious range of formalisms, including the multiple representation system (MRS) of Genesereth, Greiner, and Smith (1981), and augmented-transition networks once recommended for natural language parsing.
>
> …
>
> Closer to implementation, Grosof and Poon (2004) prototype an e-contract system called SweetDeal, based on a RuleML encoding of the knowledge representation, and a description logic representation of process ontologies. This is a relatively early example of the growing literature on computational contracting; see Surden (2012) for an overview from a legal perspective. Brammertz, et al. (2009) develop a structured formalism for financial contracts that focuses on a description of common intertemporal cash flow patterns to support securities valuation and risk analysis. This is the foundation for the Project ACTUS (2015) implementation pilot. Another recent pilot is the Financial Industry Business Ontology (FIBO), which proposes a formal, standardized model of legal concepts in finance; see OMG (2014). Still closer to implementation, the automation of accounting, valuation, risk analysis, and trade execution is a practical necessity in the daily operations of trading firms; see Brose, et al. (2014a, 2014b).

Meng Wong has undertaken a much more extensive survey of the historical landscape, which is a highly useful resource in this area. A version of his report is referenced in Appendix A to this White Paper and available on GitHub. See https://github.com/legalese/legalese-compiler/blob/master/doc/chapter-201707.org. A further summary treatment of this background by Jeannette Eicks is attached as Appendix B, and the Bibliography to this paper cites additional useful discussions.

It is worth recalling elements of the landscape that are distinct from the executable, code-embodied contracting that is the target of this initiative. First of all, there are a number of document assembly approaches, sometimes built on expert systems, sometimes incorporating model clause development. Because the intended outcome is a natural language formulation, these are on a different outcome track from the LSP project. The efforts do, however, contain useful knowledge about the underlying logic of legal expression, particularly if they are paired to an expert system for generating the contract. The structure of such an expert system can embody at least portions of the logic we are concerned with here; it simply does not end up representing it in machine executable code. As the LSP project goes forward, it will be useful to learn from the experience of the creators of such engines, such as Neota Logic, Exari, and CALI's A2J project.

Secondly, there are a number of activities seeking to mine natural language contracts and use machine learning approaches to construct typologies, do analysis of contractual holdings, and even begin to create executable options. In academia, one major ongoing project is the mirelproject.eu; previous work is described *in Semantic Processing of Legal Texts, 2010*. Commercially, startups such as Kira Systems and Luminance are bringing such approaches to market.

While often quite useful in dealing with legacy, word-based contract libraries, these approaches will have limited use for the LSP project. As discussed above at II.A, contracting is at heart a rule-generating approach, which seeks to build executable action plans with explicit, intentional components. Traditional lawyering in this area may be informed by data for purposes of conceptualization and comparison, but it builds rules. We believe that the future of contract automation will be to provide tools to contracting parties that will allow them to build rule sets with executable code rather than with words, and not in some kind of big-data enabled emulation system based on the flawed system of natural language contractual representation. We acknowledge the excitement around the potential of today's Machine Learning technologies, but we would hesitate to set our signatures to any contracts produced by such technologies.

B. Six Approaches with Broadest Impact and Greatest Relevance

Over the years, many intelligent approaches have been suggested by academia, government, non-profits, and business. Our Phase One survey to date leads us to target six particular groups of actors as being potential contributors of thought, format, and even software to the LSP project. As we finalize our survey, we invite the thoughts of our participants on these selections.

1. Expert systems: Neota, Exari, A2J, etc.

As discussed above, a number of initiatives exist that employ expert systems in contract creation and other legal drafting tasks. These structured question and answer programs are often linked with a document assembly program that produces a customized end product, generally in natural language and in traditional form. Some of these initiatives are moving in the direction of executable contracts, often linked to some kind of business process management software. To the extent that they are linked to old-fashioned text based drafting, their end product isn't useful for the LSP. What is useful, however, is the approach taken and logic embedded in the expert system itself. These can inform both the UI considerations discussed above and the underlying logic and programming requirements of an LSP. Bringing their insights on these matters into the LSP process will be useful.

Targets for consultation include:

- Neota Logic, https://www.neotalogic.com/, a for profit company. Their self-description reads, in part, "Our technology consists of an AI-powered platform and comprehensive toolset that allows professionals to rapidly build and deploy application solutions that automate their expertise, increasing productivity, improving client satisfaction and creating new business opportunities."
- Exari, https://www.exari.com/, a for profit company. Their self-description reads, in part, "Exari Contracts is the definitive Contract Lifecycle Management platform and will truly redefine the way your contracts are created, organized, and operationalized. With intuitive drafting tools and negotiation workflows, AI-driven data capture, powerful reporting capabilities, proprietary risk scoring algorithms and more, you can finally put your contracts to work for you and add incredible value across your entire enterprise."
- A2J Author, https://www.a2jauthor.org/, an initiative of CALI, a non-profit affiliated with Chicago-Kent School of Law. Their self-description reads, in part, "Access to Justice Author (A2J Author®) is a cloud based software tool that delivers greater access to justice for self-represented litigants by enabling non-technical authors from the courts, clerk's offices, legal services organizations, and law schools to rapidly build and implement user friendly web-based document assembly projects. These document assembly projects are made up of two components: a user friendly interface, called an A2J Guided Interview, and a back end template. These A2J Guided Interviews take complex legal information from legal forms and present it in a straightforward way to self-represented litigants."

2. OASIS: Legal XML, Rule ML, etc.

The development of an LSP is the natural domain for a standard setting organization. OASIS is "a nonprofit consortium that drives the development, convergence and adoption of open standards for the global information society." See https://www.oasis-open.org/. It was founded in 1993 in significant part to oversee the development of the Standard Generalized Markup Language (SGML), and since then has had a hand in standard-setting for a variety of technical needs. In the legal arena, it has sponsored a series of development initiatives around legal specification, most notably Legal XML (see http://www.legalxml.org/), and LegalRuleML (https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalruleml)

Legal XML provides data schemas for a variety of legal information. Its most developed output has been LegalXML Electronic Court Filing 4.0 (see http://docs.oasis-open.org/legalxmlcourtfiling/specs/ecf/v4.0/ecf-v4.0-spec/ecf-v4.0-spec.pdf). Although the LegalXML standards take as their starting point the filing and retrieval of legal information, and not the elements critical to legal execution, the lessons learned in this project can help to inform and refine the data standard discussed above.

The LegalRuleML approach moves further in the direction of executable representation. Its self-description provides, "The OASIS LegalRuleML TC defines a rule interchange language for the legal domain. The work enables modeling and reasoning that allows implementers to structure, evaluate, and compare legal arguments constructed using the rule representation tools provided." It is connected with a broader RuleML initiative, with applications in many domains. It seeks to provide standards for processes that combine data specification with processing architectures. In addition to the direct work of the LegalRuleML technical committee, the approach has undergirded such useful contributions as the paper *Sweet Deal: Representing Agent Contracts With Exceptions using XML Rules, Ontologies, and Process Descriptions* by Grosof and Poon (available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=442040). While the project has many sound elements, it has not yet broken through in terms of more generalized application and acceptance.

3. *Blockchain "Smart Contracts", particularly Ether/Solidity and Hyperledger*

The boom in blockchain development has included initiatives around the so-called "smart contract." Although the rather grand name would seem to imply that the LSP has already occurred, as currently employed most blockchain smart contracts are executable scripts of relatively low contractual expressivity. For the most part, they are single trigger links between some event or instruction and the delivery of cryptocurrency funds from one holder to another. As Vitalik Buterin, a leading Ethereum programmer has described it, the smart contract sets up a conditional payment instruction in code, "and the program runs this code and at some point it automatically validates a condition and it automatically determines whether the asset should go to one person or back to the other person, or whether it should be immediately refunded to the person who sent it or some combination thereof." See https://blockgeeks.com/guides/smart-contracts/ The approach is sometimes likened to a digital vending machine.

Such a mechanism is useful, corresponding to a traditional escrow account or a simple letter of credit, but it is not yet a machine executable contract of any great complexity. See https://www.technologyreview.com/s/610392/ethereums-smart-contracts-are-full-of-holes/ and https://medium.com/@philippsandner/comparison-of-ethereum-hyperledger-fabric-and-corda-21c1bb9442f6. Nonetheless, blockchain smart contracting is a step in the direction we are considering in the LSP, and the languages used to specify such arrangements may provide the basis for more the more complex expressions envisioned by the LSP. Furthermore, the current enthusiasm for all things blockchain means that there are energy, resources, and an appetite for application in this area. The initiatives which we have identified as providing opportunities for learning and exchange include:

- *Solidity* is a programming language developed specifically to match up with the Ethereum blockchain. See https://solidity.readthedocs.io/en/v0.4.21/. Its proponents declare that Solidity is capable of expressing a wide range of contractual commitments, although to date it has largely been used on the relatively limited world of "smart contracts" described above.
- *Corda* is described on GitHub as "a distributed ledger platform designed to record, manage and automate legal agreements between business partners. Designed by (and for) the world's largest financial institutions yet with applications in multiple industries. It offers a unique response to the privacy and scalability challenges facing decentralised applications." See https://github.com/corda/corda and https://www.corda.net/
- The *Hyperledger* initiative, set up under the Linux Foundation, supports a broad range of blockchain related projects, including a number of open source platforms for blockchain formation and administration. See https://www.hyperledger.org/
- The Legal Technology Labs has an initiative on Developing New Applications for Smart Contracts, leg by Cardozo's Aaron Wright. This approach largely builds on Solidity and Ethereum. See http://www.thelegaltechlab.com/index.php/developing-new-applications-for-smart-contracts

4. *Accord Project*

The *Accord Project* was announced last year. See https://www.accordproject.org/ Its goals are closely aligned with those of the LSP. "The aim is to produce a technology layer that will advance the transition to legally enforceable smart contracts by incorporating widely accepted standards relating to transactional legal practice, the use of data and distributed ledgers in contracting, and dispute resolution." See https://medium.com/accord-blog/the-accord-project-launches-industry-first-tools-and-standards-for-smart-legal-contracts-with-2e67b2b6f2fd.

It has established impressive links to other entities, working with Hyperledger, Ethereum and several other legal technology initiatives, and at least some of its roots are in the smart contracting approach discussed above. Some of its initiatives, such as Cicero, appear to be aimed at a hybrid of natural language and software expression, rather than the LSP approach of building a legal specification capability that can operate independently of natural language. See https://github.com/accordproject/cicero.

The LSP initiative will benefit from interaction with the Accord Project.

5. Business Process Management

Many of the elements that will be present in a contract standard coming from the LSP approach are already embedded in business process management software. These typically set out sequences of expected events in executable form, together with monitoring functionalities and the ability to switch to other pathways if that is warranted. The significant difference is that these systems are typically aimed at the internal operations of a single enterprise, although that characteristic can probably be modified to admit to multiple parties. An additional advantage is that there are many companies, including very significant players, offering such services, with a great deal of programming written and running. These models offer potentially fruitful points of comparison, inspiration, and code that could help to inform the LSP.

There are many providers of business process management software; here are few of them that show promise in the LSP context:

- The Object Management Group, operating in the UML (Unified Modeling Language) tradition, has produced standards such as
  - PMN: Business Process Model & Notation
  - SBVR: Semantics of Business Vocabulary and Business Rules
  - DMN: Decision Model and Notation
- As its name suggests, *Formstack* helps businesses to build smart forms. See https://www.formstack.com/ What is of particular interest for the LSP is its Conditional Logic tool (see https://www.formstack.com/features/conditional-logic). As discussed and nested processes. Formstack provides a useful point of comparison as we develop a UI to manage the creation of such statements.
- *Decisions* is "a no-code business automation platform focusing on process automation as well as data handling and business rule execution." See https://decisions.com/ It features a highly interactive user interface for designing these rules that could be a model for an LSP contract creation UI. "Decisions automates workflow processes using an integrated set of graphical tools that allow for the creation of workflows, active forms, system integrations, dashboards and reports without writing code. These designs are built and tested using visual, drag and drop - no code - design tools. Resulting apps are built in a fraction of the time of other methods - and are easier to evolve as the business requirements change."
- *IBM* is one of the larger, established companies active in this field. https://www-935.ibm.com/services/process/?lnk=mse_ts&lnk2=learn. So is *Microsoft*. See https://docs.microsoft.com/en-us/biztalk/core/business-process-management-solution. Such large players often have relatively complex, proprietary software that is coupled with contract services to design, deploy, and run large business management packages.

While such packages are often well engineered and competently e, they also are generally not designed to permit individual creation. Interoperability with other systems may also be a challenge. They may be less useful as models for an LSP approach.

- *Flora-2* and *RuleLog*, commercialized as Ergo by Coherent Knowledge, have been used to effectively model Regulation W.
- *IBM's iLog*, *Oracle's OPA*, and *Drools* are rule modeling systems.

6. Stanford's Computable Contracts Project

CodeX also houses a parallel track on computable contract development. The Computable Contracts Project is exploring a possible Contract Description Language (CDL) "designed for expressing contracts, terms and conditions, and even laws in machine-understandable way so that automated tools can be used to work with them more efficiently." See http://compk.stanford.edu/ As envisioned by this initiative, the CDL will have the following properties:

- "Machine-understandable: Computers can reason over single as well as a set of contracts: check validity, compute utility, hypothetical analysis, consistency check, planning, and execution
- Declarative & highly expressive: Specification is the program
- No need to translate domain knowledge into procedural code
- Modular: Multiple programs can be flexibly put together
- Easier to debug and visualize than procedural code
- Could also be used directly by domain experts
- There is an increasing trend of declarative approaches e.g. jQuery"

See http://compk.stanford.edu/

As with other initiatives described here, the insights gleaned from the CDL approach can contribute to the LSP project.

C. How do They Measure up to the Requirements Described Above; What Could Be Built on?

[To be developed, in part to reflect the results of our discussions at the April 6 CodeX gathering]

V. Suggestions for Next Steps

A. Suggested LSP Approach

1. Coordinated, but Decentralized and Distributed Development Process

B. Activities and Sources of Support to Date for LSP Efforts

1. CodeX, Computable Contracts

See https://law.stanford.edu/projects/stanford-computable-contracts-initiative/ and https://conferences.law.stanford.edu/compkworking201709/

2. Legal Technology Labs & Kauffman Foundation

See http://www.thelegaltechlab.com/ and http://www.thelegaltechlab.com/index.php/legal-specification-language-development

3. Others

Legalese See https://legalese.com/ and https://www.youtube.com/watch?v=qQfHe9a8zNg

A Singapore Centre for Computational Law is in the works, as a proposed collaboration between Legalese and a local university in Singapore.

Office of Financial Research See https://www.financialresearch.gov/working-papers/files/OFRwp-2015-04_Contract-as-Automaton-The-Computational-Representation-of-Financial-Agreements.pdf

Vermont Law School Center for Legal Innovation See https://www.vermontlaw.edu/academics/centers-and-programs/center-for-legal-innovation

[To be enlarged]

C. Anticipated Next Steps

1. Closure on Phase 1

White Paper Completion

- Gather comments, fill in blanks, produce next version, repeat, probably once more

Working Consensus on Technical Approach Principles

- Build on White Paper
- Build on targeted existing approaches
- Spell out working parameters as a beginning step toward a shared standard as the development process goes forward
- Small group process here

Selection of Use Cases

See list above, additional possibilities

Assemble a portfolio of natural language contracts that collectively exhibit broad coverage in terms of application and expressivity. Using this corpus, we will

- Undertake a similarity/modularity analysis
- Develop definitions/representations for states, events and triggers and sequences (trying different approaches at this stage), as you did in your paper
- Represent this as high level language (with alternatives) in the form of a mock up language (which may be paper squares on a big sheet of paper!)
- A modest analysis of expressivity versus fallibility
- Show this mock-up form in the context of the specific application area

Centers of Engagement/Teams

- Philosophy of a decentralized but coordinated process
- Activity Foci (partly territorial, partly emphasis)
    - CodeX
    - LTL
    - UCL in London
    - Others?

Funding and Staffing

- Seeking funding, both for the initiative as a whole and for particular Activity Foci and Use Cases
    - Coverage for work
    - Coverage for convening and communication
- Staffing
    - As part of existing academic or business project
    - With sufficient funding, supporting particular participants to buy time for research, building elements of the LSP, and administration and coordination

2. Moving Forward with Phase 2: Parallel Developments of Use Cases and Programming

Building the stack elements in concert with specific use cases

- Probably separate but overlapping teams across the use cases

Periodic interchange between teams to keep coordination and interoperability aspects in place

- Communications
- Meetings

Once there is progress and reasonable coalescence, then Steps 5 and 6:

5) the creation of the user interface

6) the creation of a platform on which 4 and 5 can operate

Seeking funding

- General
- For Particular Use Cases
- For Particular Centers/Teams

Planning for adoption /dissemination Activities – Phase 3

- Phase 3 aimed at dissemination and adoption will follow; planning and seeking implementation partners and resources will be an activity of Phase 2

BIBLIOGRAPHY

[TO BE DEVELOPED AND INSERTED HERE]

APPENDIX A

REVIEW OF THE FIELD I

Meng Wong has authored a extensive and authoritative review of the history of the use of computer code to represent legal specification generally and contracts in particular.   It is entitled *Computable Contracts: from Academia to Industry* and it is available at https://github.com/legalese/legalese-compiler/blob/master/doc/chapter-201707.org

APPENDIX B

REVIEW OF THE FIELD II

Jeannette Eicks has provided an additional review of the literature on Computational Contracts, drawing in part on the work of Meng Wong set out in Appendix A. This too provides useful background for the work of the LSP.

Computational Contracts: A Survey

The area of computational contracts has grown both in theory and practice in the last forty years. Since well written contracts entail deliberate precision in their creation, negotiation, execution and analysis over the period of the agreement, they are an obvious candidate for revision through computation.[7] Through computation, opportunities abound to improve the monitoring, assessing, and enforcing of contract provisions which are often not executed to the full extent specified.[8] Some attempted computational solutions address only part of the computational contracts challenge. However, whether limited in scope or an attempted synthesis of several theories, the growing sophistication of each new approach results from iterations of historic theory, research and commercial experimentation. These successive attempts to make contracts computation are an essential foundation to the eventual disruptive re-envisioning of contracts.[9] Various legal and computer science academics and professionals explored in this appendix have called for and investigated computational contracts as a potential remedy for these challenges in standard contracting processes. Experts from both subject matter areas take distinctive and often siloed approaches to making contracts computational. First we explore the highlights of some of these distinct approaches by computer science and follow with a few legal academics.

Computer scientists have approached contracts by viewing them as imprecise natural language expressions that could be specified formally through a model and/or language to provide a structure that will create efficiency, enable contract compliance checking[10] and make going to court to resolve contract disputes obsolete.[11] In his Ph.D. dissertation, Hvitved offers three categories for considering the existing work on contract formalism: (deontic) logic based

[7] Surden. Computable Contracts. p 631
[8] Lucian A. Bebchuk & Richard A. Posner, One-Sided Contracts in Competitive Consumer Markets, 104 MICH. L. REV. 827, 828 (2006).
[9] See the work of Legalese's Meng Weng Wong in gathering these legal informatics sources at https://legalese.com/v1.0/page/past.
[10] Abdelsadiq, Abubkr. "A toolkit for model checking of electronic contracts." (2013). p. 1.
[11] Tom Hvitved. "Contract Formalisation and Modular Implementation of Domain-Specific Languages". University of Copenhagen, 2014, p. 8.

formalisms, event-condition-action (ECA) based formalisms and trace based formalisms.[12] While these translations to formal languages and models are interesting scholarly works, in some cases where the contract is complex, they may be only a theoretical possibility given Chomsky's seminal work and the need to simplify in order to formalize.[13] Hvitved addresses these practical concerns through the following requirements which he suggests must be addressed within any formalized computational model:

1. Contract model, contract language, and a formal semantics.
2. Contract participants.
3. (Conditional) commitments.
4. Absolute temporal constraints.
5. Relative temporal constraints.
6. Reparation clauses.
7. Instantaneous and continuous actions.
8. Potentially infinite and repetitive agreements.
9. Time-varying, external dependencies (observables).
10. History-sensitive commitments.
11. In-place expressions.
12. Parameterized contract.
13. Isomorphic encoding.
14. Run-time monitoring.
15. Blame assignment.
16. Amenability to analysis.[14]

Hvitved's requirements list is a useful check list for evaluating the capacity of any particular formalism to represent all aspects of a contract and assures that the formalism offers a practical commercial enhancement of contracting rather than a theoretic restatement of a

[12] *Id* generally.
[13] Chomsky, N. "Three Models for the Description of Language," IEEE Transactions on Information Theory. 1956.
[14] Hvitved at 25.

single document. While most formalisms are unable to complete all of the requirements, several authors have advanced the scholarship in this area with the contribution of a wide array of contract formalisms such as deontic logics, [15] and finite automata.[16]

Pioneering work on(deontic) logic based formalisms published by Ronald Lee in the 1988 relied on viewing contracts as Petri net transition systems, where a set of states can be active at any time within the defined contract term and the actions undertaken by parties in the agreement may trigger the activation of new sets of states.[17] While Lee's approach covered most of Hvitved's requirements, it failed to gain regard due to the lack of a mapping of syntactic contracts to objects of the model.

Andrew Goodchild and his co-authors put forth a model that addresses business-to-business contracting through the use of an ECA model. In the event-condition-action (ECA) paradigm, the system waits for an event, looks for a condition to be met within that event and then applies an action. These models maintain a single rule base that is reactive to contract conditions and is enhanced through the modularity and maintainability of the contract through the reliance on an active database which manages the rule governed objects.[18] Electronic commerce depends on the creation of a dependable paradigm for expressing contracts computationally to advance the automation of the conduct of business. [19] The ECA paradigm has particular appeal to those who seek to leverage the data in existing ERP systems to inform their contract rules.[20] Similar to the deontic logic model referenced above, ECA systems acknowledge their imperfection and seek to iterate on their contractual tools as they progress.[21] Their imperfection comes through the lack of well-defined grammar and lack of a semantic accounting of their language as well as their rather exclusive focus on business-to-business contracts.[22] The Goodchilds et al paradigm, while insightful and more lawyer friendly than other models, lacks completeness.

---

[15] Ronald M. Lee. A Logic Model for Electronic Contracting. Decisions Support Systems, 4(1):27 -44, 1988.
[16] Mark A. Flood and Oliver R Goodenough. "Contract as Automaton: The Computational Representation of Financial Agreements." Office of Financial Research Working Paper, March 27, 2017. Accessed at https://www.financialresearch.gov/working-papers/files/OFRwp-2015-04_Contract-as-Automaton-The-Computational-Representation-of-Financial-Agreements.pdf on May 20, 2017.
[17] Lee at 27 -44.
[18] Papamarkos, George, Alexandra Poulovassilis, and Peter T. Wood. "Event-condition-action rule languages for the semantic web." Proceedings of the First International Conference on Semantic Web and Databases. CEUR-WS. org, 2003.
[19] Boulmakoul, Abdel, and Mathias Sallé. "Integrated contract management." Proceedings of the 9th Workshop of the HP OpenView University Association. 2002. p 13
[20] Goodchild, Andrew, Charles Herring, and Zoran Milosevic. "Business Contracts for B2B." ISDO 30 (2000).
[21] Kimbrough, Steven O., and Dongjun Wu, eds. "Formal modelling in electronic commerce." Springer Science & Business Media, 2006.
[22] Goodchild, Andrew, Charles Herring, and Zoran Milosevic. "Business Contracts for B2B." ISDO 30 (2000).

The expansive grammar and policy language that some considered a flaw in the Goodchilds et al approach was revised through a model based on normative statements put forth by Boulmakoul and Sallé.[23] The Boulmakoul and Sallé approach begins with deontic logic principles and generates a more compact language that yields a formalization whose structure is highly reflective of the original contract.[24] Another ECA approach is the business contract language (BCL) of Milosevic et al..[25] The BCL seeks to monitor event based business activities and thus aligns itself with a workflow management approach to contracts. The language and architecture (the business contract architecture or BCA) are provides the capacity to provide a full enterprise contract management solution for the extended enterprise focusing on express behavior and constraints.[26] Through this ECM, contract monitoring would be enhanced, though many aspects of computational contracting would be overlooked. Later work by Milosevic and Governatori focused on the formalization of BCL for the purpose of representing the violation of contractual obligations and the valuation of penalties surrounding those violations temporally in a run-time environment.[27] Both ECA approaches reflect business driven approaches to formalization that translate more directly into application development and an iterative process.[28]

The functional programming approach taken by Peyton Jones focused on less complex bilateral financial contracts.[29] By limiting the scope of the contract type they model, Peyton Jones is able to demonstrate the possibility of describing and valuing a large class of contracts, such as bundled derivatives or other groups of financial agreements.[30] He introduced a combinatory library in Haskell that allows the description of such financial contracts and combinational denotational semantics that valuate these contracts. Peyton Jones concedes that what they constructed is a mundane use of "a functional language to define a domain-specific library, thereby creating an application-specific programming language."[31] However, "from the standpoint of financial engineers, our language is truly radical: they acknowledge that the lack of a precise way to describe complex contracts is "the bane of our lives."[32] The analytic capacity and descriptive ability of this approach are its strengths, while the narrow financial

[23] Boulmakoul, Abdel, and Mathias Sallé. "Integrated contract management." Proceedings of the 9th Workshop of the HP OpenView University Association. 2002. p 13
[24] *Id*.
[25] Milosevic, Zoran, et al. "On design and implementation of a contract monitoring facility." Electronic Contracting, 2004. Proceedings. First IEEE International Workshop on. IEEE, 2004.
[26] *Id* at 8
[27] Governatori, Guido, and Zoran Milosevic. "A formal analysis of a business contract language." International Journal of Cooperative Information Systems 15.04 (2006): 659-685.
[28] Of course, iterative processes are fine when the contract is short term and will be regenerated in a short period. However, if the contracting method keeps evolving,
[29] Jones, Simon L. Peyton. "Composing contracts: An adventure in financial engineering." FME. Vol. 2021. 2001.
[30] *Id.*
[31] *Id* at 2.
[32] *Id* at 2.

focus of the exercise weakened the usefulness of their approach outside of financial markets. The combinatory library would need to be greatly expanded to encompass contracts more broadly, outside of the world of finance.

Andersen et al. consider another approach, narrowly tailored to illustrate the usefulness of their process algebra model. Andersen et al. restrict their model to consider only contracts that govern the exchange of money, good and services between parties.[33] In this model all transfers are considered events that are assessed for fulfillment based on a trace-based semantic model that sets out a sequence of necessary and finite events which are required for successful contract completion.[34] In this system contracts either succeed or fail with no capacity to evaluate which party failed and how to remedy that failure.

Finally, in 2017 Farmer and Hu propose a formal language for writing contracts, noting that in a world of increasing complexity where contract are naturally becoming more complex "continuing to write complex contracts in natural language is not sustainable if we want the contracts to be understandable and analyzable."[35] Farmer and Hu postulate that dynamic contracts whose actions taken by parties, events that happen, and values fluctuate over time dependent on certain temporal observations which they reference as *observables*.[36] Farmer and Hu's FCL language includes formal semantics and a dependence on simple type theory logic which grants their language access to higher-order logic, function types, quantification over functions, function abstraction and concise notation forms.[37] Furthermore, Farmer and Hu's conception of FCL includes a reasoning system that can specify an expression for a satisfied agreement, violated agreement, defunct rules, fulfilled contract and breached contract as well as simulate those aspects of the contract for any time specified.[38]

While computer scientist progressed with formal representations of contractual arrangements to help resolve ambiguous, unnecessarily complex contracts,[39] legal scholars have argued that ambiguity can serve a purpose,[40] complex documents cannot and, perhaps, should not be dramatically simplified[41] or that computable contracts will broaden the need for contract law.[42]

[33] Andersen, Jesper, et al. "Compositional specification of commercial contracts." International Journal on Software Tools for Technology Transfer (STTT) 8.6 (2006): 485-516.
[34] *Id*.
[35] Farmer, William M., and Qian Hu. "A Formal Language for Writing Contracts." Information Reuse and Integration (IRI), 2016 IEEE 17th International Conference on. IEEE, 2016. Updated January 10, 2017. http://imps.mcmaster.ca/doc/fcl.pdf *last accessed* May 20, 2017.
[36] Id at 5.
[37] *Id* at 5.
[38] *Id* at 14.
[39] Need Cite – plenty in computational contract literature.
[40] Fortgang, Ron S., David A. Lax, and James K. Sebenius. "Negotiating the spirit of the deal." Harvard Business Review 81.2 (2003): 66-79.
[41] Surden, Harry. "Computable contracts." UCDL Rev. 46 (2012): 629.

Some legal professionals have maintained that programmatic contracts would be brittle and laden with bugs[43] while others have suggested that contract complexity renders contract inscrutable to computers and difficult to manage over the course of their lifecycle.[44] Without question, legal scholars understand the value proposition that historic EDI (Electronic Data Interchange) and more current block-chain style smart contracts bring to businesses.[45] In his ground-breaking article, Surden acknowledges the value of data-oriented contracts expressed in a highly structured way.[46] He also asserts that the reliance on data is necessary given computers' inability to parse natural language and the limitations of mathematically based languages to express the rich contextual aspect of terms found in contracts.[47] He concludes that while computable contracts may reduce transaction costs, they are rigid and "not all, or even most, contractual arrangements or aspects of contracting are amendable to the data-oriented and computable paradigm."[48]

Werbach and Cornell take up the discussion of computable contracts through the more focused lens of block-chain driven smart contracts with an emphasis on the execution layer of agreements.[49] The Werbach and Cornell article distinguishes smart contracts from the area of contract law, stating that "while smart contracts can meet the doctrinal requirements of contract law, they serve a fundamentally different purpose."[50] Their premise is that contract law is a remedial institution that is not focused on ensuring performance *ex ante*, but rather resolving through justice practices grievances that arise *ex post*.[51] They state that smart contracts function by making remediation unnecessary by admitting no possibility of breach.[52] This drawing of the line between the contract as an instrument and the area of law that surrounds the instrument helps clarify the important and continuing role of legal professionals in contract law.[53] Werbach and Cornell end their look at smart contracts by warning that "machines are prone to their own errors and biases" and will create challenges of legal and practical accountability.[54]

[42] Werbach, Kevin D., and Nicolas Cornell. "Contracts Ex Machina." (2017) forthcoming 67 Duke Law Journal at p 6.
[43] Martin, Kingsley. "Contract Analysis and Contract Standards: Developing CAD for Law—Part 2 Contract Building Blocks." 21 May 2011. Contract Analysis and Contract Standards. Web at http://contractanalysis.blogspot.com/2011/05/ last accessed May 20, 2017.
[44] Surden, Harry. "Computable contracts." UCDL Rev. 46 (2012): 629.
[45] Werbach, Kevin D., and Nicolas Cornell. "Contracts Ex Machina." (2017) forthcoming 67 Duke Law Journal.
[46] Surden.
[47] Surden at 642.
[48] *Id* at 700.
[49] Werbach, Kevin D., and Nicolas Cornell. "Contracts Ex Machina." (2017) forthcoming 67 Duke Law Journal.
[50] *Id* at 4.
[51] *Id* at 4.
[52] *Id* at 4.
[53] *Id* at 5.
[54] *Id* at 54.

The above introduction to the area of computational contracts has spans forty years and is by no means exhaustive. More concrete examples of computational contracting efforts may be found within the computer scientists' theoretical writings than those of legal scholars. Generally the approach of computer scientists has been toward offering part of a solution while legal scholars speculate that only a piece of the contractual legal puzzle will be solved by computational law.   Both agree that the deliberate specificity of contracts both in their creation and execution encourage the consideration of fully computational instruments. However, the approaches vary and all currently fall short of a complete solution to computational contracts. Until a complete solution with easy application is broadly available, computable contracts will never become a mainstream piece of legal practice. The eventual disruptive re-envisioning of contracts will draw its architecture from a combination of thirty years of prior knowledge and approaches. In this world where the placement of an oxford comma may cost millions,[55] the costs of non-computational contract complexity have been shown to lead to contract incompleteness,[56] ambiguity and poor drafting may lead to a disastrous misreading of obligations,[57] contracts often look like spaghetti code,[58] and consumers demand increased efficiency and cost effectiveness, current contracting processes may be expedient business triage rather than the formation of a standardized, strong and viable legal solution. The legal industry awaits the right, fully complete, single computational solution to the creation of all contracts.

[55] Victor, Daniel. "Lack of Oxford Comma Could Cost Maine Company Millions in Overtime Dispute." New York Times. N.p., 16 May 17. Web. 16 May 17. <https://www.nytimes.com/2017/03/16/us/oxford-comma-lawsuit.html?_r=0>.
[56] Anderlini, L. & Felli, L.. "Incomplete Contracts and Complexity Costs." Theory and Decision February 1999, Volume 46, Issue 1, pp 49.
[57] Darmstadter, Howard. "Precision's Counterfeit: The Failures of Complex Documents, and Some Suggested Remedies." The Business Lawyer 66.1 (November 2010): 61-83, p.65. JStor. Web. 14 May 2017.
[58] Horstmann, Cay (2008). "Chapter 6 - Iteration". *Java Concepts for AP Computer Science* (5th ed. [i.e. 2nd ed.]. ed.). Hoboken, NJ: J. Wiley & Sons. pp. 235–236. Phrase *originated here*: Conway, Richard (1978). *A primer on disciplined programming using PL/I, PL/CS, and PL/CT*. Winthrop Publishers.