Developing a Legal Specification Protocol:

Technological Considerations and Requirements

LSP Working Group

February 14, 2019

CONTENTS

A Note on References

The references set out in this white paper are not in customary legal citation format. Rather, they are, for the most part, in a version of scientific reference form. In some cases, where we believe there is no loss of specificity or of understanding in what is being pointed to, the reference has been made simply to a web-based URL. These choices may be justified partly because only limited reference is being made to traditional legal sources and partly because in the age of URLs and the migration of authority from print to electronic resources adherence to the older formatting should not be slavishly required. Perhaps inconsistently, the white paper has retained the custom of legal scholarship to use footnotes as a vehicle for many of the references, and have not provided a separate bibliography. A future version of this white paper may rectify this omission. The footnote approach does have the advantage of putting the details on the reference close to the text to which they related. Criticism for these decisions should be leveled only at this paper's principal author, and should not be imputed to any other contributor or to Stanford's CodeX program as a distributor of this paper.

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.[1] These initiatives gave some service, but the full flowering of the Internet as a platform required establishing the Internet Protocol Suite, an architecture and a set of shared conventions for representing and transmitting data that could be used and built upon by a number of different applications.[2]

We are at a similar crossroads in legal automation – 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. For example, in the United States "Blue Book Form" creates a shared standard for classifying print-based citation data.[3] 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 to develop an expressive Legal Specification Protocol (LSP) 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

---

[1] See, e.g.,. Vaughan-Nichols, Steven J., 2015. "Before the Web: Online services of yesteryear," available at https://www.zdnet.com/article/before-the-web-online-services/ ; Garling, Leb, 2012. "Before the Web, AOL, and Prodigy, There Was MINITEL," Wired, available at https://www.wired.com/2012/06/services-begone/

[2] A brief but detailed history of the development of the Internet is available at Leiner, Barry M., Cerf, Vinton G., Clark, David D., Kahn, Robert E., Kleinrock, Leonard, Lynch, Daniel C., Postel, Jon, Roberts, Larry G., & Wolff, Stephen, 1997. *Brief History of the Internet*, available at https://www.internetsociety.org/internet/history-internet/brief-history-internet/. See, generally, *International Journal of Internet Protocol Technology,* http://www.inderscience.com/jhome.php?jcode=IJIPT

[3] "The Bluebook" at https://www.legalbluebook.com/Public/Introduction.aspx.

new needs. Interoperability[4] (the ability to work across many platforms and formats) and generativity[5] (the openness to embody work not anticipated when created) 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, even though a number of private actors are developing useful pieces of the system (see the discussion at Section V, below). An LSP is therefore a good target for development through a collaborative process involving private sector actors, academic centers, philanthropy and governmental standard setters. As described in Section VI below, the initiative out of which this white paper has emerged has consisted of a loose network of just such actors – our working group.

As will be discussed more fully in Section V below, the LSP project can rely on steps already taken in projects such as OASIS' Legal XML, CALI's A2J Author, Accord, Open Law, 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, aiming at a specific use case or piggybacking onto natural language. And most haven't been designed with the full expressivity of broad-based legal specification in mind. Finally, none have achieved momentum for broad adoption that is a basic necessity for an LSP. The time has come for a more holistic design approach that can provide a widely adoptable standard – which may include significant elements from these existing approaches.

B. Project Stages

The project should be developed in stages. The key steps for this initiative involve:

1) specifying the functional requirements 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 steps #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;

---

[4] E.g., Palfrey, John, and Gasser, Urs, 2011. *INTEROP: The Promise and Perils of Highly Interconnected Systems*, Basic Books.

[5] E.g., Zittrain, Jonathan, 2011. *The Future of the Internet and How to Stop It*. Yale University Press, also available at http://futureoftheinternet.org/download/; Goodenough, Oliver R., "Generativity: Making Law a More Open Institutional 'Ecosystem' for Productive Innovation" (April 2, 2015). Vermont Law School Research Paper No. 4-15. Available at SSRN: https://ssrn.com/abstract=2589263 or http://dx.doi.org/10.2139/ssrn.2589263

5) creating or promoting the creation of the user interface;

6) creating or promoting the creation of a platform on which 4 and 5 can operate; and

7) disseminating the protocol and, to the extent developed the interface 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 summarizes work to date on the first phase, and provides 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. 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 use 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, with several nodes of activity in communication with each other. In this we are seeking a pathway that we believe better suits 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. Contracts present a relatively tractable problem: they are typically shorter, more discrete legal formulations than legislation or regulation, while still containing 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 less challenging than adopting 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.

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 III.B of this paper, we are currently targeting several specific contract categories for this purpose.

D. This Paper

This paper is intended to sum up the conclusions of the working group on steps #1 and #2 above.

In defining the requirements, this paper will first explore the uses and functionalities of traditional word-based contracting. Delving deeper, it will suggest how we may 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 summarizing 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. In this discussion we will address the degree to which defeasibility and deontic logic are necessary components in an LSP. We follow with a survey of some of the activities to date in the legal specification field, with an emphasis on contracts. As part of this survey we identify specific areas and initiatives that we believe need particular attention as we develop the LSP. Finally, we outline a plan of action for going forward.

II. Focus on Contracts and What They Do

A. The Functionality of Contracting

To be useful in creating contracts, an LSP must represent the things that contracts do. There are many approaches to theorizing contracts.[6] Functionally, a contract provides an actionable outline for committed performance in a multi-party relationship. The specified actions can be affirmative: things a party promises to do. The specification can also create boundaries: things that 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 traditional financial contract, a new pathway kicks in, providing for notice, possible cure, acceleration of obligations, rights to seize collateral, and other new and different actions and limits. The tripwires that trigger these new pathways can either link directly to the expected performance or to important ancillary

[6] A useful introduction to these concepts is set out in Turocy, Theodore L. and von Stengel, Bernhard, 2001. "Game Theory," available at http://www.cdam.lse.ac.uk/Reports/Files/cdam-2001-09.pdf. For greater detail, see, e.g., Schecter, Stephen & Gintis, Herber, 2016. *Game Theory in Action: An Introduction to Classical and Evolutionary Models,* Princeton University Press.

facts, events, or actions, through mechanisms such as "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.[7] 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.[8]

B. The Techniques of Contracting

The classic techniques of contracting connect to the functions and challenges described above. Natural language statements typically provide the formalism for describing the events and outcomes that constitute the commitment structure of the contract. The starting point for most contracts is a statement of 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 cases, this statement of the expected path of performance may appear in an addendum prepared by the business people who will carry 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

[7] E.g., Goodenough, Oliver R., 2008. "Values, Mechanism Design, and Fairness" in *Moral Markets: The Critical Role of Values in the Economy*, Paul J. Zak, ed. Princeton University Press, also available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=933012

[8] E.g,, Id; Engel, Christoph & Schweizer, Urs., 2008. "Mechanism design and the law." *Journal of Institutional and Theoretical Economics JITE*. 164. 1-3. 10.1628/093245608783742273; Güth, W. 2017. "Mechanism Design and the Law" in Parisi, Francesco, ed., *The Oxford Handbook of Law and Economics: Volume 1: Methodology and Concepts*, Oxford University Press.

discussed by Grosof and Poon (2004).[9] This structure can be observed in practice in the ISDA Agreement that underpins most of the derivative and swaps financing markets.[10]

Software representations of portions of these processes already exist. A firm's business process management software can often manage much of the "happy path" of expected action. Whether aimed at manufacturing and delivery or at financial calculations and payment, these steps have already been a significant target for automation. For example, Amazon's one-click ordering puts such automation into practice,[11] as do most so-called "smart contracts" in the cryptocurrency world (see Section V.B.5 below).

Some contracts emphasize boundaries or permissions over positive actions. For instance, a distribution contract for a film may limit the rights granted to specified 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. Violation of a constraint may trigger a default cascade that could include cancellation, damages, or other remedies.

More complex contracts can spend a great deal of attention on the definition of the circumstances that invoke an alternative pathway. Such a switching event is often termed a "default." In classic contracting, triggering events can create default in three different contexts.

The first, and most clear-cut, is a 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 return the contract back into its expected pathway. Lacking cure, 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.

Second and third, a complex contract can specify a set of ancillary performance criteria ("covenants"), and/or ancillary facts and states of the world ("representations and warranties") that also act as switch-points. For instance, in a loan contract, the borrower may promise to stay current on her tax payments. This covenant does not relate directly to the payment of interest and principal to the lender, which may still be flowing smoothly. Tax delinquency, however, potentially changes the risk profile of the transaction; the lender may want to accelerate repayment when the borrower faces garnishment. Likewise, a change in the tax law, unrelated to any action by the parties, could shift the nature of the bargain and trigger the applicability of a new track.

---

[9] Benjamin N. Grosof & Terrence C. Poon (2004). "SweetDeal: Representing Agent Contracts with Exceptions Using Semantic Web Rules, Ontologies, and Process Descriptions," *International Journal of Electronic Commerce*, 8:4, 61-97, DOI: 10.1080/10864415.2004.11044305

[10] ISDA, 2002. *2002 ISDA Master Agreement*, available at https://www.isda.org/book/2002-isda-master-agreement-english/

[11] "Why Amazon's '1-Click' Ordering Was a Game Changer," *Knowledge @ Wharton* (2017), available at http://knowledge.wharton.upenn.edu/article/amazons-1-click-goes-off-patent/

A great deal of lawyering in complex contracts manages the covenants and representations and warranties, together with the consequences of a breach of one of these ancillary boundaries. These lead naturally to further negotiation over how to enforce the parties' obligations, whether under the expected deal or under a different set of default-induced terms. Sometimes, automatic consequences can be built in, such as 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 backstop.

C. Lawyers Constructing Contracts

How do lawyers construct contractual instruments that embody this functionality? They typically modify existing templates for constructs of event and outcome; sometimes, they must draft new "legalese" formulations. Legalese, the complicated and stilted language of contracts, regulations and other legal documents, uses natural language to specify events (or connected strings of events) and their consequences. Logical structures, such as if-then-else conditional statements then chart the expected "happy path" and the default-linked alternatives.

Lawyers often attack such drafting problems intuitively, without expressly understanding the process they are undertaking.[12] One of the side benefits of examining the computational structure of contracts is that it helps a practitioner clarify what is going on, even without actually encoding these statements into software. One of the requirements for a useful contract specification protocol will be a user interface that will allow the easy construction of such structures.

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

Discussions of legal automation often point out that word-based specification of contracts and other legal formulations can contain "ambiguity" or partial definitions that leave matters open to some interpretation.[13] Lawyers sometimes view this as a desirable design element and suggest that clarification of ambiguity is a barrier to contract automation. 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.[14] We emphasize the distinction between "rule ambiguity"

---

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

[13] Other terms used to describe this property (or its near relations) include "uncertainty," "vagueness," or "discretion." Other scholars have argued with some persuasiveness that there are distinctions that can be captured by the differences between these and other similar terms. Pursuing these distinctions is beyond the scope of this white paper and is not necessary for understanding the broader point made here.

[14] Looking at law more broadly, some point to common law reasoning, the process of extracting decision principles from existing fact patterns and their legal consequences, as providing a zone of discretion for the decision maker that can be a desirable feature in law. See, e.g., D'Amato, Anthony, 2010. "Legal Uncertainty,"

and "event ambiguity." Rule ambiguity means the pertinent facts are well established, but the contract does not clearly specify the resulting outcome. Most drafters – and their clients – would view rule ambiguity as a flaw. 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. It is a pervasive problem in computing more generally, occurring prominently in areas such as feature extraction, sensor design, and channel coding. For example, consider facial recognition software: Is the person viewed really Kim Kardashian or someone else? If a Kardashian, implement Plan A, if not, use Plan B. The 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.

"Ambiguity" sometimes refers to the intentional use of events that require human judgment to resolve. A contracting party will use "reasonable efforts" or will 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 ambiguity, so much as delegation to a human "black box" to determine the event. Once the authoritative human(s) (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, and exists in the "slow AI" of traditional law. In blockchain parlance, the human or committee of humans is an "oracle."[15] AI judges are NOT a necessary component in a computationally assisted and specified legal system.[16]

E. Links to Legal Specification More Generally

Although we have emphasized contracting in the LSP project to date, we believe that an approach to enable contract specification and execution can also support legal specification

---

Faculty Working Papers. Paper 108, available at http://scholarlycommons.law.northwestern.edu/facultyworkingpapers/108

[15] E.g. von Kohorn, Doug, 2018. "Blockchain Oracles Will Make Smart Contracts Fly," *Hackernoon*, available at https://hackernoon.com/oracles-help-smart-contracts-resolve-subjective-events-d81639d8291c

[16] 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 Nobel laureate Oliver Hart summarized at Hart, Oliver, 2017. "Incomplete Contracts and Control," *American Economic Review* 2017, 107(7): 1731–1752, https://doi.org/10.1257/aer.107.7.1731, and available at https://scholar.harvard.edu/files/hart/files/incomplete_contracts_and_control.pdf . A more comprehensive discussion is not within the purview of this white paper.

more broadly. This is particularly true for the explicit-rule formulations generally found in legislation and regulation, and which are the main modes of rule articulation in the civil law system.[17] 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.[18] In this context, our efforts may be broadly conceived as a Computational Restatement of the Law of Contracts.

III. Use Cases

A. Use Case Goals and Stages

The development of a widely used LSP cannot take in the abstract. 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 securely anchored in particular cases. The goals and stages of working with the use cases reflect the following stages:

A *first stage* that 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 that 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 isolate 20 distinct events, ranging from time passing to going bankrupt, as constituting the information that a short financial contract will potentially consume in its life cycle, including a limited set of default triggers and consequences.[19]

We will take the targeted use-case contracts and perform a similar listing of salient events. We will then break those events apart further into their components of time, measurement,

---

[17] See, e.g., the summary provided by The Robbins Collection of the Boalt Hall School of Law, University of California at Berkeley, "The Common Law and Civil Law Traditions," https://www.law.berkeley.edu/library/robbins/CommonLawCivilLawTraditions.html. For more in depth treatment, see, e.g., Apple, James G. & Deyling, Robert P., 2012. *A Primer on the Civil-Law System*, Federal Judicial Center, available at https://www.fjc.gov/sites/default/files/2012/CivilLaw.pdf

[18] See generally "About ALI," available at https://www.ali.org/about-ali/

[19] M. Flood, and O. Goodenough, 2015. "Contract as Automaton: The Computational Representation of Financial Agreements," OFR Working Paper, No. 15-04, March 2015, available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2648460

performance, and conclusions from other reasoning process. The goal is to identify the minimum taxonomy of content types necessary to 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 IV.B of this white paper.

On the processing side, we need to identify the logical connectors necessary to express legal/contractual reasoning, as well as connectors that can powerfully bundle the concepts. For instance, if-then-else may be adequate 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 of many steps in a fully expanded if-then-else representation to relatively few steps in a more powerful logic system.[20] 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, a 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, moving 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.

B. Identifying Use Cases

At this stage we have targeted several possible use cases, and we are open to adding a limited number of additional targets to our list. Selection criteria include:

- Domain knowledge in our working group
- Availability of partners 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

[20] If-then is a core feature of all logics, and in that sense cannot be dispensed with. The point of these "shortcuts" is not to avoid if-then, but to deploy it in architectures that permit greater compression and avoid a full iteration of extensive logic trees at each stage of the process.

- Outcome importance, which can be judged by several criteria, including social utility, promotion of justice, financial importance, and potential impact.[21]
- Funding opportunities to support use case development.

Applying these factors, initial targets are as follows:

- Financial contracting,
- Telecoms service level agreements,
- Early stage investment,
- Manufacturing supply chains,
- Construction contracts, and
- International trade.

Several of these have potential links to blockchain applications; such an involvement would be a desirable element in the mix. Blockchain systems have catalyzed computational contracting, and should be a component in the development of the LSP. We welcome additional use cases meeting the criteria set out above and in particular providing some further diversity of subject matters. A detailed description of each of these is not included in this white paper; developing and selecting the use cases will be an immediate task of the next stage of the LSP project.

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

In defining a computational protocol for the law, we identify two foundational elements: a data standard and a logic standard. The data standard must be able to represent, with adequate structure and interoperability, the details about the world that law digests and emits for its operations. The logic/programming standard must be able to state and process the logical connectors to manage the event/consequence chains that compose legal logic. The standard should also be able to this parsimoniously. Representation of all-important tasks should be possible (theoretical expressivity) but representation of common tasks should be easy (practical expressivity).[22]

In addition to these core standards, acceptance of the LSP will likely depend on the ability to link it to one or more user interfaces that will permit non-coders to construct relatively complex legal formulas in code through intuitively appealing representations. These links include both

---

[21] These factors helped to inform the Use Cases Working Group at the 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, available at https://goo.gl/xzbKxr

[22] Farmer, William (2007). "Chiron: A multi-paradigm logic". In R. Matuszewski; A. Zalewska. *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*. Studies in Logic, Grammar and Rhetoric. pp. 1–19. ISBN 978-83-7431-128-1.; for background, see also "Expressive power (computer science)," *Wikipedia*, https://en.wikipedia.org/wiki/Expressive_power_(computer_science)

input, as from sensors, prior proceedings, or other sources, and output, both as further data and as instructions to execution systems.

As graphical interfaces made computing widely accessible when they replaced DOS style command code entries, so too will good UI approaches turn computational law from a theoretical possibility for the rare coder/lawyer composite to an everyday tool for courts, legislatures, regulatory bodies, members of the bar, and non-professionals. Such an interface is the expectation in most user marketplaces, and the LSP user community will expect no less.

This section will focus first on logic requirements, then data approaches, and finally on user interfaces.

A. Logic and programming of legal event and consequence

Over the years, many logics have been suggested for legal representation.[23] While one can derive shortcuts and subroutines from some of the more complex logics, for the most part, current legal drafting (including contracts) reduces to relatively simple conditional structures, such as if-then-else statements.[24] The highest level of complexity is generally limited to terms such as "if, but only if." This kind of conditional reasoning creates a natural language "pseudo code" we call legalese.

Although laws, contracts and regulations can be *very* complex, this complexity often results from a string of relevant events/facts that is quite long, strung together with many and/or and related connectors. These are often cumulated and linked together in "chained conditionals." Although humans often have trouble tracking and parsing such chains, these building blocks are generally elements of a recognized fact universe coupled with simple if-then switches. It is the volume of inputs and switches that creates complexity, and not the logic itself.

One way to make this more tractable is to break the chains down into repeatable components, providing nested sub-routines and objects. In the natural language world, a computational object can be framed as a defined term, used in several places, or as a cross-reference to the computational statement in a different part of the law. As with object-oriented programming, such defined points used repeatedly cut down on recursions in the statement of the rule.

For example, consider the defined term "Accredited Investor" in 17 CFR 230.501(a) as part of Regulation D of the U.S. Securities Laws. The definition is relatively complicated, with eight subcategories applying differently to companies, people, non-profits, trusts, etc. Once met, however, the category is used extensively throughout Reg. D, and in many other state and

---

[23] See, e.g., the survey in Prakken, Henry & Sartor, Giovanni, 2002. "The Role of Logic in Computational Models of Legal Argument: A Critical Survey," in A.C. Kakas, F Sadri eds., *Computational Logic*, LNAI 2408, pp. 342-381, 2002

[24] For an introduction to if/then/other logic, see, e.g., the material at Code.org, "Lesson 9: "if-else-if" and Conditional Logic," *CS Principles 2017*, https://curriculum.code.org/csp-1718/unit5/9/#if-else-if-and-conditional-logic1

federal laws and regulations. None of these subsequent uses restate all eight categories; rather, the definition allows incorporating the computation of whether an investor is "accredited" into many other computational contexts. The specification of the eight subcategories itself is composed entirely of involved strings of simple conditional reasoning.

The legalese specifications of contracts (and legal rules more generally) also nest up against other layers with implied application, such as the Uniform Commercial Code,[25] which provides gap-fillers, boundaries, and other supplements and constraints on commercial contracting in the U.S., or as the rules of court procedure if enforcement requires litigation.[26]

One further wrinkle is the utility of "defeasible logic" in legal specification. There are a number of different treatments of this concept.[27] In the approach described by Governatori:

> A defeasible theory is a structure D = (F,R,>) where F is a set of facts, represented by literals, R is a set of rules, and >, the superiority relation, is a binary relation establishing the relative strength of rules. Thus, given two rules, let us say r1 and r2, r1 > r2 means that r1 is stronger than r2, thus r1 can override the conclusion of r2.[28]

Under this defeasible approach, rather than defining an exception to a more general rule as a "but if" step in a long logic tree, the two rules are run simultaneously and the exception is given priority of outcome to the general rule – is "stronger" in this sense. Because the law is full of exception style provisions, utilizing defeasibility and its multi-path structure in the logical architecture of legal specification can provide significant compactness to the processing of particular legal rules. That said, if recursions are permissible, most if not all of what defeasibility can do can be captured in a long enough string of conditional statements. Most natural language legal formulas take some version of these approaches – a partial explanation for why legalese formulations can get so lengthy. Individual humans are not good at

---

[25] The Uniform Commercial Code is available at https://www.law.cornell.edu/ucc; for a summary of the development and purpose of the UCC, see, e.g., Steingold, David M. 2013. "What is the UCC?," NOLO, available at https://www.nolo.com/legal-encyclopedia/what-is-the-ucc.html

[26] *Federal Rules of Civil Procedure*, available at https://www.law.cornell.edu/rules/frcp

[27] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher, 2000. "A flexible framework for defeasible logics." In *Proc. American National Conference on Artificial Intelligence* (AAAI-2000), pages 401-405, Menlo Park, CA, 2000. AAAI/MIT Press, available at http://espace.uq.edu.au/eserv.php?pid=UQ:9714&dsID=acis1999.pdf; Antoniou, Grigoris, Billington, Guido Governatori, and Maher, Michael J. "Embedding Defeasible Logic into Logic Programming Theory and Practice of Logic Programming," in, pp 803-835. Cambridge University Press, available at http://www.governatori.net/papers/2011/icail2011carneades.pdf; Grosof, Benjamin N. & Poon, Terrence C., 2004. "SweetDeal: Representing Agent Contracts with Exceptions Using Semantic Web Rules, Ontologies, and Process Descriptions," *International Journal of Electronic Commerce*, 8:4, 61-97, DOI: 10.1080/10864415.2004.11044305; and Kowalski, RA & Sadri, F., "Logic programs with exceptions," in *New Generation Computing* 9 (3-4), 387-400. See, generally, the resources at http://www.governatori.net/research/pubs/defeasible.html

[28] Governatori, Guido, 2011. "On the Relationship between Carneades and Defeasible Logic," available at http://www.governatori.net/papers/2011/icail2011carneades.pdf

simultaneous parallel processing, and so the Governatori architecture is unavailable in natural language formulas.

In a different strategy, the defeasing conditions can be set out early in the statement, so that the branching takes place before extensive computation has occurred. If one can put the exception conditions early in the tree, it may be possible to limit their impact on the complexity of any given stage in the computation.

The key upshot is that modern law consists of complex statements (e.g. the Tax Code) built out of extensive terms applying relatively simple logic in highly involved ways. There may be a place in the future for using more complex logics in legal specification, but simple logic, using conditional statements and supplemented with defeasibility, should be sufficiently expressive for the work of the LSP.[29]

A lucky aspect of the law's use of if-then logic employing conditional statements of this kind is that it can be represented in a wide range of existing computer languages and approaches. Because of this capability, we probably don't need a specific new legal representation language per se as part of the LSP, although special approaches may eventually be useful for creating wider ranges of legal specification. Languages aimed specifically at contracts and law include Solidity[30] and Legalese (used here as the name of a programing language, and not in the more common sense of complex natural language legal specification).[31] General computer languages with the necessary conditional capability include Python,[32] Java,[33] JavaScript,[34] and C++,[35] just to name a few. If we start with these, we will need to develop sub-versions that can efficiently represent the specific kinds of tasks that law generally and contracts specifically require. Within these versions, we will quickly accumulate libraries of code to perform the common structural steps of contracts and laws more generally, such as the default structure of many contracts. And we will 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.

By bringing examples such as Solidity and JavaScript into the discussion we do not intend to simply jump prematurely into implementation. As we develop an LSP focus on representing the core domain abstractions. A key contribution of the LSP should be to *constrain* the set of things

---

[29] The use of the term "conditional statements" here is not meant as a reference to the programing approach sometimes described as "conditional logic". See, e.g. http://blog.ruleml.org/post/32629706-the-sad-state-concerning-the-relationships-between-logic-rules-and-logic-programming/

[30] See, e.g., https://solidity.readthedocs.io/en/v0.4.24/

[31] See https://legalese.github.io/aboutus

[32] See, e.g., https://opentechschool.github.io/python-beginners/en/conditionals.html

[33] See, e.g., https://www.lynda.com/Java-tutorials/Programming-conditional-logic/377484/421324-4.html

[34] See, e.g., https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/conditionals

[35] See, e.g., https://cal-linux.com/tutorials/conditionals.html

that can be represented, adapting more general languages to the somewhat limited scope of legal event/consequence specification.

A secondary benefit of the LSP will be to expose legal formulations to automated tests for completeness or contradiction. Just as "every non-trivial program has at least one bug," most contracts and regulations will contain a few such mistakes. Automated checking will help good lawyers get even better at their drafting.

Some argue that legal specification requires "deontic logic."[36] This is a form of philosophical reasoning that is specially adapted to moral statements of obligation, such as what one "ought" and "should" do. In this respect, it is critical to the definition of norms. There is a useful debate on the utility of deontic approaches to representing the law. For the limited case of contracting, however, current, word-based law, such as statutory, regulatory and contract prose, clearly *avoids* language of moral obligation and instead uses direct statements of event and consequence. More generally, law operationalizes norms by taking abstracted aspirations of obligation and creating specific formulations of event and consequence. Even relatively imprecise formulations, such as those embedded in common law court determinations, lead to set consequences when triggered. Such formulations reset the game structure to make desirable behavior more likely. A contract differs from an aspiration, and a law differs from a moral precept, in the link through consequential logic of the deed and the result.

This approach finds support in "Bad Man" theory of law put forward by Oliver Wendell Holmes, Jr.[37] In his 1897 speech "The Path of the Law," Holmes distinguished between the promptings of morality and the hard consequences of legal rules. He argued:

> If you want to know the law and nothing else, you must look at it as a bad man, who cares only for the material consequences which such knowledge enables him to predict, not as a good one, who finds his reasons for conduct, whether inside the law or outside of it, in the vaguer sanctions of conscience. The theoretical importance of the distinction is no less, if you would reason on your subject aright.

Human psychology deploys deontic logic in conceptualizing norms. Human law deploys consequential logic in specifying outcomes. Deontic logic is a fascinating object for study and thought, but it will not be necessary for most exercises in representing legal outcomes in machine executable terms. That said, deontic approaches may have a role to play in designing

---

[36] Navarro, Pablo E. & Rodriguez, Jorge L., 2014. *Deontic Logic and Legal Systems*, Cambridge University Press (Cambridge Introductions to Philosophy of Law); Woleński, Jan, 2016. "How deontic logic contributes to the analysis of legal systems" *Revus*, available at http://journals.openedition.org/revus/3518 . See, also Kowalski, Robert, 2017. "Satisfiability for First-order Logic as a Non-Modal Deontic Logic," available at https://www.doc.ic.ac.uk/~rak/papers/satisfiability.pdf . Bob Kowalski suggests that attempts to represent legal documents in LPS show that that deontic syntax (not semantics) is a natural way for users to understand and express preferences between goals and intentions.

[37] Oliver Wendell Holmes, Jr., 1897. "The Path of the Law," 10 *Harvard Law Review* 457 (1897)

an effective interface for human users, who have a predilection for thinking in deontic terms. This will be explored in section IV.C below.

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.[38] 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.

To enable greater automation in legal specification and analysis, it will be critical to create a widely used convention for first describing events that are consumed and produced by computational law and second describing the conclusions of the legal process, whether as information for a further process or as instructions for some kind of action. These needs exist both in the specific use case of contracts as well as in the broader case of legal specification generally. A standard that captures these elements is a necessary component in the "stack" of any widely shared and sufficiently expressive legal specification protocol.

We summarize below a set of elements that meets the informational requirements of competent legal automation approaches. The adoption of conventions for 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 distills a relatively comprehensive taxonomy of the salient information requirements for the predicates and outputs of the law's if-then-else computational processes. This taxonomy aligns with prior work by the ESTRELLA project expressed in the Legal Knowledge Interchange Format (LKIF), 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 emerged independently.

---

[38] See, e.g., Flood & Goodenough, supra note 19. Robert Kowalski and others have suggested that a further category, "fluents" can be useful in understanding the relation of law and computing. See, e.g., Kowalski, Robert & Sergot, Marek, 1986. "A Logic-based Calculus of Events," *New Generation Computing* 4(1986): 67–94. # 1986, available at https://www.doc.ic.ac.uk/~rak/papers/event%20calculus.pdf. See, generally, the references at https://scholar.google.co.uk/citations?user=pvkQAj0AAAAJ&hl=en#d=gs_md_cita-d&p=&u=%2Fcitations%3Fview_op%3Dview_citation%26hl%3Den%26user%3DpvkQAj0AAAAJ%26citation_for_view% . In the automata formalism, these fluents may represent intermediate states that can be worth labeling as points of return.

i. A statement of the event/data type

a. Can reference dictionary, taxonomy, ontology, etc.

b. Can reference a natural language description, but need not do so

c. Distinguishes between information and instruction

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

c. If an instruction, is the contents of that instruction

iii. Provenance/Source

a. Can be a sensor, a blockchain record, a court determination, the product of a prior computation, etc.

iv. Time/Date stamp

a. Can state in universal time

b. Relates back to provenance (this item could be a sub-field of iii)

c. Distinguishes event time and report time

v. Matter

a. A particular contract, court case, application, legal citation, etc.

vi. Specification of the event in other systems (aimed at creating interoperability making the standard 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 Will be Developed.

We are not, at this time, suggesting a particular architecture for specifying these elements. Such a formal representation is a goal of the LSP initiative, not its starting point. Some kind of

articulated character string seems a likely candidate. Other domain-linked specifications, such as XML, can provide a point of departure for working out the architecture once the necessary/desirable elements are agreed upon.

The LSP architecture can also assist in structuring databases to store the standard elements of the LSP representation. For instance, in relational databases, the structure of the database itself provides elements of the eventual processing around that data.[39] These considerations should inform the design of the LSP information standards. The more sophisticated ontologies discussed below, such as LKIF, embody some of these properties.

3. Legal and Financial Ontology Approaches

The development of a widely shared standard for legal information and instruction can draw on several initiatives involving legal and financial ontologies.[40] These range in complexity from a lexicon of standardized natural language terms describing commonly encountered events, facts and concepts with legal relevance, to deeply considered data structures.[41]

The law makes use of a number of natural language ontologies. Almost any complex statute or regulation has a system of definitions that act as a primitive ontology. Other specialized semantic frameworks are more elaborate. For example, the standard Incoterms terminology for shipping and international trade created by the International Chamber of Commerce is in wide use and has been adapted to local law in most major trading jurisdictions.[42]

The Legal Knowledge Interchange Format (LKIF), is noteworthy among proposed schemas for computable legal representation. LKIF is a principal initiative of the European Union's ESTRELLA project, whose principal goal is a set of interoperable standards for the exchange of legal

---

[39] See, e.g., the summary at Mullins, Craig S., 2012. "Database Administration: Creating the Database Environment," *INFORMIT*, available at http://www.informit.com/articles/article.aspx?p=1963781&seqNum=4, excerpted from Mullins, Craig S. 2012, *Database Administration: The Complete Guide to DBA Practices and Procedures, 2nd Edition*, Addison-Wesley Professional. The foundation of the relational database approach is set out at Codd, E.F., 1970. "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM* 13:6 p. 377-387, available at https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

[40] Casellas, Núria, 2010. *Legal Ontology Engineering: Methodologies, Modelling Trends, and the Ontology of Professional Judicial Knowledge*, Springer. See also, Hoestra, Rinke, Breuker, Joost, Di Bello, Marcello, and Boer, Alexander, 2009. "LKIF Core : Principled Ontology Development for the Legal Domain," in *Frontiers in Artificial Intelligence and Applications*, 188(1):21-52, available at https://pdfs.semanticscholar.org/3ace/023a84a215245af526167d6f5bca0416945f.pdf; Casanovas, Pompeu, Sartor, Giovanni, Biasiotti, Maria & Barrera, Meritxell, 2011. "Introduction: Theory and Methodology in Legal Ontology Engineering: Experiences and Future Directions." In G. Sartor, P. Casanovas, M.A. Biasotti, M. Fernández-Barrera, eds., *Approaches to Legal Ontologies. Theories, Domains, Methodologies. Law, Edition: Law, Governance and Technology Series, vol. 1*, Springer Verlag.

[41] See, generally, Roussey, Catherine, Pinet, Francois, Kang, Myoung Ah, & Corcho, Oscar, 2011. "An Introduction to Ontologies and Ontology" in G. Falquet et al., *Ontologies in Urban Development Projects, Advanced Information 9 and Knowledge Processing 1,* Springer, available at http://oa.upm.es/10381/1/An_Introduction.pdf

[42] See "Shipping Terms: What are Incoterms?" *Trade Finance Global,* https://www.tradefinanceglobal.com/freight-forwarding/incoterms/

information.[43] ESTRELLA also developed the more basic MetaLex standard.[44] Summarizing the LKIF approach:

> LKIF has two main roles: enable the translation between legal knowledge bases written in different representation formats and formalisms and secondly, as a knowledge representation formalism that is part of a larger architecture for developing legal knowledge systems.[45]

LKIF is based on the Web Ontology Language (OWL), the widely used standard for creating and structuring formal ontologies.[46]

The LegalXML initiative sponsored by OASIS is also worth noting.[47] This effort includes the LegalXML eContracts Version 1.0 Committee Specification aimed at contract specification.[48] The LegalXML approach has had limited uptake, including adoption by the Brazilian Legal and Legislative information Portal.[49]

In the financial sphere, the Financial Industry Business Ontology (FIBO) is a joint project of the Object Management Group and the Enterprise Data Management Council.[50] FIBO draws on OWL principles to create an ontology of financial industry terms, many with salience to contract law specifically as well as to legal specification as a whole.

More generally, Schema.org provides approaches to creating interoperable information across the web.[51]

We anticipate that all of these approaches will inform the LSP project's information and data standards. Some, but not all, are aimed more at data retrieval tasks rather than at event-based computability, and so their utility as models may be somewhat limited.

---

[43] "ESTRELLA," *European Standardization Organizations*, https://www.cencenelec.eu/standards/Education/JointWorkingGroup/Documents/Standardization_in_ResearchInnovation_SuccessStories_IT_ESTRELLA.pdf

[44] Broer, Alexander, Winkels, Radboud, & Vitali, Fabio, 2008. "Proposed XML Standards for Law: MetaLex and LKIF" in Pompeu Casanovas, Giovanni Sartor, Núria Casellas, & Rossella Rubino, eds. *Computable Models of the Law, Languages, Dialogues, Games, Ontologies,* available at https://pdfs.semanticscholar.org/0fed/4f7f194d50b0a356a31ab6eb52a169b10532.pdf

[45] Hoekstra et al., note 40 supra, at 43

[46] Hitzler, Pascal, Krötzsch, Markus, Parsia, Bijan, Patel-Schneider, Peter F., Rudolph, Sebastian, 2012 *OWL 2 Web Ontology Language Primer (Second Edition),* W3C, https://www.w3.org/TR/2012/REC-owl2-primer-20121211/

[47] OAISIS LegalXML, *OASIS*, http://www.legalxml.org/about/index.shtml

[48] OASIS LegalXML eContracts TC, *OASIS*, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalxml-econtracts

[49] All in one place: Brazilian Legal and Legislative Information Portal, *LexML*, http://projeto.lexml.gov.br/documentacao/resumo-em-ingles

[50] EDM Council, "FIBO Ontology," available at https://spec.edmcouncil.org/static/ontology/; Object Management Group, "Financial Services Standards," available at https://www.omg.org/hot-topics/finance.htm

[51] See https://schema.org/

4. Utility of a Data Standard for Analytic Approaches

A critical side-benefit of a widely-shared standard for legal information and instruction will be the creation of structured data sets that support not just in the computation itself, but also other data collection and analysis tasks. For instance, a bank whose loan and investment portfolios were expressed in machine-executable terms could run stress tests under different scenarios and then analyze the resulting data. By expressing data consistently, we facilitate comparison, aggregation, and analysis. Not only will a widely shared LSP achieve its primary goal of deploying machine computation on the specification and execution of legal rules, it will also enable a broad range of analysis on the workings and outcomes of the legal system.

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

The LSP will need to interact effectively with both humans and other machine processes. We consider these separately.

1. Interacting with Humans

Sitting on top of the LSP, a human-facing user interface (UI) should provide a relatively intuitive construction space for creating the chains of event and consequence that are the typical output of legal specification. While no special logic may be necessary for a great deal of legal/contractual specification, presenting the logic for intuitive use and construction by contract drafters will require careful design.

At this stage in the project, we need to identify more fully the processes and approaches that humans use to construct legally useful chains of event and consequence to help delineate design criteria. Questions include the following:

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

While the law may not require the moral precepts of deontic logic, they may provide human cognitive shortcuts that we can build on in creating an appealing UI.

In addition to the human elements underlying UI design, we also need to delineate in more detail the processes and particulate components of contractual relations at a level of generalization one or two steps up from the simple conditional statements that can embody contracts. Take the frequent contracting element known as an "affirmative covenant" as an example. This is a category of promises made in traditional Anglo-American contracting that

contain obligations ancillary to those at the core of the contractual performance itself. A useful interface to such an approach must address questions such as:[52]

- 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/non-performance 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 employ a symbolic, graphical interface, if not a drag and drop interface similar to Scratch or to Decisions in the business process management world.[53] Prior art includes the Business Process Model and Notation (BPMN) and Semantics of Business Vocabulary and Business Rules (SBVR) initiatives of the Object Management Group (OMG), Contract-Oriented Diagrams, and work on visual contracts by Helena Haapio and Stefania Passera.[54]

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

- Input of key events or facts, including sources for the information;
- Workflow-style and the sequences of common tasks, including if-then-else constructions;
- Management of outputs, including export functionality; and
- Major switch points and their triggers

The ultimate goal is a user-friendly design space for "programming" legal documents, with the capacity to export output for execution by both humans and machines.

2. Interacting with Other Machine Technologies

A principal advantage of the LSP will be the ability to collect and provide legally relevant data and outcomes directly through integration with other technologies even more broadly than

---

[52] For a discussion of the issues, see John J. Camilleri, Gabriele Paganelli, and Gerardo Schneider, A CNL for Contract-Oriented Diagrams (2014), available at https://arxiv.org/pdf/1406.5691.pdf.

[53] See https://scratch.mit.edu/ and https://decisions.com/.

[54] On the OMG initiatives, see https://www.omg.org/bpm/. On contract-oriented diagrams, see Camilleri, et al., supra note 52. On visual contracts, see Passera, Stefania, Haapio, Helena & Curtotti, Michael, 2014. "Making the Meaning of Contracts Visible – Automating Contract Visualization," in Erich Schweighofer et al. (Eds.) *Transparency. Proceedings of the 17th International Legal Informatics Symposium IRIS 2014*.; Jusletter IT, 20 February 2014. Available at SSRN: https://ssrn.com/abstract=2630609

current systems can achieve. On the input side, this could be as simple as linking up with a sensor, such as thermometer, to take in relevant information, or as complex as an interface with a business process software package. On the output side, conclusions from a contract could activate payments, or the shipments of goods. It could also trigger other legal processes, or enforcement actions. Interactivity with other technologies and technologically-based processes will be as important as interactivity with humans.

Making such interaction possible will involve data, communication and security standards. The data needs have been discussed above. Communication standards, such as those contained in the Internet protocols, should facilitate the easy transmission of data across a variety of platforms and mediums. The design of a widely shared protocol for messages supporting legal computation should provide for interoperability with the likely communication channels.

The security layer may provide domain-specific challenges. Legally relevant data and instructions will often have significant consequences, and may be tempting targets for hacking and falsification. The internet protocol has evolved to support several security techniques, such as encryption, firewalls, etc. The LSP will need to incorporate core security as well as being open to supporting increasingly powerful layers of additional protection.

3. Interacting with Blockchain and Distributed Ledger Technology

A point of particular importance will be the ability of computational contracts and other machine-executable legal specifications to interact with blockchains and other distributed ledger technologies. The current "smart contract" initiatives, explored more fully below, have emerged largely in the blockchain context. Although their use has focused on relatively simple payment transactions, they have the potential for much more complex specification. While there will be many LSP applications that will be independent of distributed ledger technology, a widely shared LSP will need to "play well" with blockchains.

Additionally, distributed ledgers are likely to play an important role in the security layer. The design of the protocol should keep in mind the needs of blockchain recordation and execution, whether through a hashing approach or otherwise.

V. Existing Technological Efforts

A. General

There is a long history of work on the representation of legal statements in executable code. Flood and Goodenough summarize:[55]

---

[55] Supra, note 19. Meng Wong has also provided a useful summary of much of the historical work in the area at "70 Years of R&D", *Legalese*, https://legalese.com/ and in a version appearing as "Computable Contracts: from Academia to Industry," Subchapter 9.1, in Stephan Breidenbach & Florian Glatz, 2018, *Rechtshandbuch Legal*

> 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).

Most expert system approaches are distinct from the executable, code-embodied contracting that is the LSP's target. First, several document-assembly approaches, sometimes built on expert systems, incorporate 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 representation 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 UI characteristics relevant to the LSP, even if machine-executable code is not an output. 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, all discussed in V.B below.

Second, there are several activities seeking to mine natural language contracts and use machine learning approaches to construct typologies, analyze contractual holdings, and even begin to create executable options. In academia, one major ongoing project is the MIREL project (see

---

Tech, C.H. Beck, available at https://www.beck-shop.de/breidenbach-glatz-rechtshandbuch-legal-tech/productview.aspx?product=20343998

mirelproject.eu); previous work is described in *Semantic Processing of Legal Texts,* (2010).[56] Startups such as Kira Systems and Luminance are commercial examples.

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 fundamentally 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 the traditional approach still builds rules.

We believe that the future of contract automation will be to provide tools that allow contracting parties to build rule sets with executable code rather than with words; this will not be a big-data emulation based on the flawed system of natural language contractual representation. Although machine learning has much promise, including guiding users of an LSP, in the near future we would hesitate to set our signatures to contracts of any complexity produced directly by such technologies without some kind of human review.

B. Six Approaches with Broadest Impact and Greatest Relevance

Many groups in academia, government and industry have proposed thoughtful approaches to computable legal representation. We highlight here six particular groups as potential contributors of thought, format, and even software to the LSP project. This list is necessarily incomplete even as we promulgate it, and it will become increasingly dated as initiatives enter and leave the space. Nonetheless, it contains our current working targets, and the next phase of the LSP should evaluate each of these approaches more fully. At the very least they will provide useful design insights; one or more of them may also provide the basis for some or all of the LSP. Future phases in the LSP process should be open to this possibility.

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

A number of initiatives employ expert systems in contract creation and other legal drafting tasks. These structured question-and-answer programs often integrate with a document assembly component to produce a customized end product, generally in natural-language form.[57] Some of these initiatives are moving in the direction of executable contracts, often

---

[56] Francesconi, E., Montemagni, S., Peters, W., Tiscornia, D. (Eds.) 2010. *Semantic Processing of Legal Texts: Where the Language of Law Meets the Law of Language*, Springer. Available at https://www.springer.com/us/book/9783642128363

[57] Because the output of these systems is typically a projected action command based on a determined system of inputs and outputs, it can be argued that they embody "imperative logic" rather than "declarative logic." Because there is no possibility for choice, no consequence for failure to choose is necessary, and without such consequence it isn't a legal statement of the kind we have been exploring here. See, e.g., Kowalski, Robert, "Logic Programming in the 1970s," in Cabalar P., Son T.C. (eds) *Logic Programming and Nonmonotonic Reasoning*. LPNMR 2013. Lecture Notes in Computer Science, vol 8148. Springer. These examples are useful nonetheless as providing sources of insight both on the structure of the event space leading to the commands and on the user interfaces which they have developed to assist in relatively intuitive design. Furthermore, to the extent that computational

linked to business process management software. Although conventional text-based drafting isn't useful for the LSP, the UI design and steps embedded in the expert system itself can inform the underlying logic and programming requirements of an LSP.

Targets for consultation include the following:

- Neota Logic, https://www.neotalogic.com/, a for-profit company. "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.

  > 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.

  > 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

---

contracting results in direct commands to a business process, as envisioned in IV.C above, then the boundary between declarative and imperative logics will become permeable.

founded in 1993 mostly 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 legal specification development initiatives, most notably LegalXML (see http://www.legalxml.org/), and LegalRuleML (https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalruleml)

LegalXML 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/legalxml-courtfiling/specs/ecf/v4.0/ecf-v4.0-spec/ecf-v4.0-spec.pdf). Although the LegalXML standards take as their start with 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.

LegalRuleML 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, co-founder of the Ethereum platform, has described it, the smart contract sets up a conditional payment instruction in code (see https://blockgeeks.com/guides/smart-contracts/):

> … 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.

The approach is sometimes likened to a digital vending machine. [58]

Such a mechanism corresponds 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 work of Open Law (https://openlaw.io/), with leadership from Aaron Wright, is taking this to a new level of sophistication, and may also provide an anchor for a more generalized LSP.

The initiatives which we have identified as providing opportunities for learning and exchange include the following:

- *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 open source platforms for blockchain formation and administration. See https://www.hyperledger.org/
- Open Law and the Legal Technology Labs have an initiative on Developing New Applications for Smart Contracts, led by Cardozo's Aaron Wright. This approach largely builds on Solidity and Ethereum. See https://openlaw.io/
- The Agreements Network, a project linked to Monax (see https://monax.io) also seeks to provide aa technical platform for the automation of previously bespoke legal services. See https://agreements.network
- The Pact-Smart-Contract Language, linked to the Kadena blockchain, seeks to correct flaws that its exponents perceive exist in Solidity. See https://kadena.io/docs/Kadena-PactWhitpaper.pdf

---

[58] Szabo, Nick, 1997. "Formalizing and Securing Relationships on Public Networks" available at https://nakamotoinstitute.org/formalizing-securing-relationships/. See also, Goodenough, Oliver R., supra note 7.

4. *Accord Project*

The goals of the very active *Accord Project* (https://www.accordproject.org/) align closely 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.

Accord 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.

5. Business Process Management

Many of the elements that will be present in a contract standard coming from the LSP approach exist already 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. A 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. As with expert systems, these business managements systems are often "imperative," i.e. just hard-wired links of input and output, with little consideration of more complex conditional concerns.[59] Legal contracting – and its automation – can be useful for designers in this context, because it is a mechanism for coordinating *across* managerial domains.

There are many companies, including very significant players, offering business process management services, with a great deal of software running in production contexts. These models offer potentially fruitful points of comparison, inspiration, and code that could inform the LSP:[60]

- The *Object Management Group* (https://www.omg.org/), mentioned earlier, 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

---

[59] See the discussion at note 29 above.

[60] For a useful paper on an approach to the Semantics of Business Vocabulary and Rules (SBVR), see Mark H. Linehan, SBVR Use Cases, IBM T.J. Watson Research Center, available at https://pdfs.semanticscholar.org/5e7c/92ac4eb1b85945ca75acfd40a2a3626c2b44.pdf

    - DMN: Decision Model and Notation
- As its name suggests, *Formstack* helps businesses to build smart forms. See https://www.formstack.com/ Of particular relevance for the LSP is Formstack's Conditional Logic tool (see https://www.formstack.com/features/conditional-logic) 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 active in this field, https://www-935.ibm.com/services/process/?lnk=mse_ts&lnk2=learn, as is *Microsoft; s*ee https://docs.microsoft.com/en-us/biztalk/core/business-process-management-solution. Such large players offer complex, proprietary software coupled with contract services to design, deploy, and run large business management packages. While such packages are often well-engineered, they also do not generally support 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/

The Worksheets project is a specific application growing out of Stanford that allows the creation of active forms that can perform contract functions. See http://worksheets.stanford.edu/homepage/index.php and https://conferences.law.stanford.edu/compkworking201709/wp-content/uploads/sites/40/2017/07/Worksheets-CompK.pdf. Commercialization of the approach is happening through https://www.symbium.com/.

As with the other initiatives described here, the insights gleaned from the CDL approach and the specific Worksheets application can each contribute to the LSP project.

VI. LSP Next Steps

A. Suggested LSP Approach

In the context of this white paper we do not seek to create a specific project proposal, with budgets, timelines, etc. Rather, we seek to set out a framework for considering what such proposals should seek to accomplish, and some general thoughts on how we – or others – might proceed.

1. Coordinated, but Decentralized and Distributed Development Process

So far, the LSP has been carried forward by an informal group of universities, researchers and application developers, under the loose sponsorship of Stanford's CodeX Center for legal informatics and with support from the Ewing Marion Kauffman Foundation. We suggest the continuation of this approach, with the development of additional "nodes" internationally, described below, and with an expanded basis of financial support. We do not believe that the LPS is ripe for any more formal governance. Rather, the next stages can be carried forward by coordinated activity across a loose network, with sufficient communication and periodic "gatherings" to share developments and mediate disagreements. At a future date, a more formal structure may be appropriate.

Section I.B above envisions seven broad stages for the LSP project. The key steps for this initiative cluster into phases:

Phase I

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 Steps 1 and 2, establishing use cases for focusing our efforts; and seeking contributions of labor and of financial backing for the next steps;

Phase II

4) building on Steps 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 Steps 4 and 5 can operate; and

Phase III

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

This white paper has described conclusions drawn from the work to date on steps 1 and 2 of the first phase. It also provides a bridge to beginning the work on the second.

B. Anticipated Next Steps

1. Closure on Phase 1

As we close out phase one, this white paper provides conclusions on steps 1 and 2. In wrapping up this phase, we anticipate an additional LSP white paper providing more detail on the history of existing projects and a report on the projected use cases.

With respect to step 3, through the attendance at our large meetings we have identified a network of potential collaborators. While there is additional work to be done in enlarging and reinforcing this network, we have a foundation for this going forward.

Building on this work, we are ready to engage in taking the project forward into the second phase: steps 4, 5 and 6.

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

In the second phase, we anticipate taking the following actions as our starting points:

a. Assembling 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)
- Represent this as high-level language (with alternatives) in the form of a mock up language
- Undertake an analysis of expressivity versus fallibility
- Show this mock-up form in the context of the specific application area

b. Building the stack in concert with specific use cases:

- Creating the core elements of software, data standards, and user interface that support the needs specified above in this white paper
- Selecting use cases considering the criteria described above and using them to focus and test the development
- Probably separate but overlapping teams engaged across the selected use cases

c. Structuring periodic interchange between teams to keep coordination and interoperability aspects in place:

- Communications
- Meetings

Once there is progress and reasonable coalescence on these points, then the initiative will tackle Step 6, the creation of a platform on which the stack can operate.

3. Funding

The success of this phase will more than ever depend upon funding to support the work itself, interchange events that will keep the various teams operating, and a light, but nonetheless critical, layer of organization and coordination. We anticipate raising funding for the following:

- General use to cover the costs of organizing and coordination
- Specific use cases, which may be linked to interested players in the targeted field
- Centers and teams, which may be linked to particular government or other location-specific funding

The amounts vary, but are likely to total seven figures in U.S. dollar terms. Raising sufficient funding will be a key element in the success of Phase 2. This work is in progress with a number of targets, and there is reason to be hopeful that, spread across the various nodes, the necessary funding can be deployed.

4. Planning for Phase 3

A final element in Phase 2 will be planning the adoption and dissemination activities that are the core of Phase 3.

VII. Thanks and Acknowledgements

Oliver Goodenough is the principal author of this white paper. There have also been direct contributions and invaluable editorial input from John Cummins, Jeannette Eicks, Mark Flood, Benjamin Grosof, Bob Kowalski, Rebecca Purdom, Susan Salkind, Harry Surden, Roland Vogl and Meng Wong.

Many individuals and institutions have contributed to the progress we have made on Phase 1 of the LSP project. Properly thanking the many attendees at the large meetings alone would fill several pages. Accepting the risks of *inclusio unius*, we should acknowledge the following:

Institutions:

- CodeX: the Stanford Center for Legal Informatics
- Legal Technology Laboratory
- The Ewing Marion Kauffman Foundation
- The Gruter Institute for Law and Behavioral Research
- The Office of Financial Research of the United States Treasury
- The Vermont Law School Center for Legal Innovation
- University of Colorado Law School

Individuals (alphabetically):

- Evan Absher
- Monika Cheney
- John Cummins
- Mary Beth Dewey
- Michael Genesereth
- Oliver Goodenough
- Guido Governatori
- Benjamin Grosof
- Mark Flood
- Tony Lai
- Rebecca Purdom
- Susan Salkind
- Harry Surden
- Roland Vogl
- Meng Wong