Computational Law

Nathaniel Love
Stanford University
353 Serra Mall
Stanford, CA 94305
natlove@stanford.edu

Michael Genesereth
Stanford University
353 Serra Mall
Stanford, CA 94305
genesereth@stanford.edu

ABSTRACT

Computational law is an approach to automated legal reasoning focusing on semantically rich laws, regulations, contract terms, and business rules in the context of electronically-mediated actions. Current computational tools for electronic commerce fall short of the demands of business, organizations, and individuals conducting complex transactions over the web. However, the growth of semantic data in the world of electronic commerce and online transactions, coupled with grounded rulesets that explicitly reference that data, provides a setting where applying automated reasoning to law can yield fruitful results, reducing inefficiencies, enabling transactions and empowering individuals with knowledge of how laws affect their behavior.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*relation systems, representations*; F.4.1 [**Theory of Computation**]: Mathematical Logic—*computational logic*

1. INTRODUCTION

The application of computational logic to legal domains came early in the development of the field of artificial intelligence and law, as researchers embraced the concept that by formally representing laws in logic, computers could be used to process, apply, and analyze these laws [16, 9].

Unfortunately, these ideas have not achieved their full potential. Edwina Rissland notes that, "Law is not a matter of simply applying rules to facts via *modus ponens*" [12], and when regarding the broad application of AI techniques to law, this is certainly true. The rules that apply to a real-world situation, as well as even the facts themselves, may be open to interpretation, and many legal decisions are made through case-based reasoning, bypassing explicit reasoning about laws and statutes. The general problem of open texture when interpreting rules, along with the parallel problem of running out of rules to apply when resolving terms, presents significant obstacles to implementable automated rule-based reasoning [17]. Also, in many legal domains, the facts of a situation themselves may be unclear or incomplete: human intervention and interpretation is necessary to make these facts available to a legal reasoning system so that it can even apply the rules. This further adversely effects usability and any notion of correctness. To combat these shortcomings, some rule-based systems have been hybridized with case-based systems, or augmented with meta-rules or with nonmonotonic, defeasible reasoning techniques, in order to make them more suitable for general applications in law [13, 15, 5].

However, there are domains in which explicit rules govern behavior, and the actions taken by agents in those domains can be directly monitored by automated systems: electronic commerce, internal business processes, and other electronically-mediated transactions taking place online. In these domains, individuals, companies, and government agencies act under the scope of wide-ranging sets of rules, including federal, state, and local laws, contracts, business rules, and other self-imposed policies. Here, semantically-rich, electronically-mediated transactions (orders, reservations, financial transactions, and other web services) are ripe for inspection by automated systems, because the execution of specific actions by various agents can be precisely observed. When machine-processable versions of laws, regulations, business rules, and contracts are all brought to bear on governing electronically-mediated behavior, *computational law* systems that represent and reason about those diverse rulesets in conjunction with agent activity can do what rule-based systems in general legal domains cannot.

Electronic domains make the two central problems of rule-based interpretation of laws manageable. First, in computer-mediated domains, the movement toward semantic data makes the facts of a situation widely and readily available, and these facts can be agreed upon by all involved parties: actions are logged, with times, amounts, and involved agents explicitly noted. Second, the rules that govern these domains can be made significantly easier to interpret: machine-processable rules can unambiguously reference the semantically rich and concretely observed facts. Of course, this requires either that current textual representations be converted to a machine-processable form (not without significant issues of interpretation), or that new constraints (business rules or contract terms, e.g.) be written directly in electronic form. In either case, the benefits of increased efficiency and transactions, enabled by the availability of computational law systems, provide incentives to move to

ICAIL '05, June 6-11, 2005, Bologna, Italy.

machine-processable rules in parallel with the adoption of semantic data.

Computational law systems take advantage of the direct, semantic connection between the terms in laws, rules, and regulations, and the model of the digitally-mediated world manipulated by agent actions, enabling a rigorous, formal approach to legal reasoning. Unlike some other legal reasoning applications, the setting for computational law systems is not in the courts, but rather at the level of the individual or business interacting with the complex rules of a larger enterprise: these entities can use computational law to plan, execute, and monitor transactions within the governing enterprise. Computational law systems can enable transactions and increase efficiency, although when conflicts arise, resolution may still need to take place outside of the system.

2. COMPUTATIONAL LAW

A representation language for computational law must enable processing of both semantic data and multiple, semantically rich rulesets in the context of a formal model of behavior. The need for knowledge representation and automated reasoning calls for a language based in logic, especially one with a capability of representing and reasoning about sequences of agent actions, as well as behavioral constraints– laws –on these actions. While ontology development lifts semantic content out from the unstructured web, enabling computers to reason about static data, a unified logical language for encoding machine-comprehensible rules enables computers to reason about complex behaviors and transformations of that data. The techniques of computational logic, applied to the semantic rules as well as the data, form the basis of a computational law system.

2.1 Enterprise Modeling

A setting for computational law requires an *explicit* model of an enterprise, fully instantiated and implemented online. Enterprise modeling formalisms (UML, Petri nets, state machines) link semantic data with action definitions that explicate how that data changes in response to the behavior of agents acting within the enterprise. With this foundation, the enterprise model can then be expanded with constraints on agent behavior that are grounded in the fundamental, semantic constructs of the enterprise.

2.2 Rule Repositories

Many applications for computational law systems require reasoning about overlapping regimes of rules, from multiple jurisdictions of governmental regulations to combinations of contracts and business rules. In single domains governed by a specific set of rules, experts have hard-coded laws into form-based systems that automatically reason about user-entered data (especially tax, where programs like TurboTax[7] see wide usage). The computational law model generalizes to any electronically-mediated domain governed by arbitrary sets of rules: as a spreadsheet enables data processing with respect to arbitrary mathematical formulas, computational law systems can reason about agent behaviors with respect to any set of logically encoded business rules, governmental regulations, or contract terms. Enterprises that govern agent actions can establish rule repositories, creating a regulatory network that user applications can draw from to assist in planning, monitoring, and executing transactions. Government agencies can publish regulatory rules, while businesses can establish intranet repositories for internal business rules, workflow specifications, and contracts from which their employees can extract relevant rulesets for their activities.

2.3 Legal Self-Help

Supplied with semantic data and rules, computational law systems can empower individuals with legal self-help: understanding how laws and regulations govern their behavior, and assisting in structuring and planning electronic transactions to bring about individual goals. While certain activities necessarily require the assistance of attorneys, many transactions complicated by overlapping, complex sets of rules can be clarified and executed with the assistance of a computational law system. Like the word processing programs that freed users from requiring typesetters for document preparation, computational law systems can assist users in structuring transactions that are valid with respect to complex behavioral constraints, without the need for outside legal assistance. This framework does not replace legal counsel, especially in cases of conflict, but rather reduces transaction costs and increases efficiency in the mundane, everyday tasks that still must be executed under regulatory regimes. Computational law is not geared toward the courthouse setting: "self-help" is not intended to mean *pro se* (as in [4]).

Legal self-help covers the basic example of an individual understanding the constraints on their behavior within an enterprise, but computational law systems should support operations beyond these planning and monitoring tasks. Enterprise administrators may want regimented computational law systems, that actively prevent agents from performing certain behaviors; they may want to use computational law to help understand potential rule changes within the enterprise, or even to construct such rules automatically to meet certain goal conditions (the automated legislation/contract formation task).

3. EXAMPLE

To motivate our approach to the formalization of computational law, and to explicate the desired operations of computational law systems, we use the example setting of a university, where a myriad of individuals operate within an environment with complex rules.

Within this environment, university-wide policies dominate the regulatory regime, although schools and departments within the university structure their own rules governing course prerequisites, space usage within their buildings, and other areas unspecified by university regulations. The hierarchical structure of these rulesets mirrors the layers of federal, state, and local regulations that govern transactions outside the private setting of the university campus. A computational law system for the university demands a hierarchy of semantically rich rule repositories: in order to perform a transaction, individuals need assistance in assembling the correct sets of rules that constrain the execution of the transaction, considering their positions within the university as well as the nature of their activities.

The university setting is already a largely electronically-mediated domain, as course registration, room reservations, coursework submission, and many other transactions take place through electronic transactions. In order to apply a

computational law system to this domain, however, the university's data must be both integrated and semantic, and individual actions must take place through web services that produce semantically interpretable effects: the applications used by individuals are conversant with the rule repositories as well as semantic data. When assisting a student in registering for a final semester of courses, the computational law system automatically applies data from that student's transcript to simultaneously resolve both departmental course prerequisite rules and university breadth requirements. This activity is an instance of legal self-help, as the computational law system serves to mediate the student's interaction with a complex governing body, helping to plan and execute a transaction legal with respect to the university's rules.

While for certain individuals, the university acts as a governing body, the domain also shares the characteristics of a corporation, where employees need to consult business rules, contracts, and outside governmental regulations in guiding their activities. For this setting, a computational law system needs to support dynamic rule repositories, and provide assistance in determining the ramifications of changes to these rulesets. A departmental administrator can make semantically rich modifications to room reservation policies (for example, making a certain room only available to tenured faculty), and individual users can immediately see the impact of these constraints on their existing planned events. An administrator in one building, short on meeting space but rich in equipment, could establish a *quid pro quo* contract with an administrator in a spacious but equipment-poor building, agreeing on rules that dictate when and how members of one department can reserve resources from the other department. This contract, then, forms an additional entry in the rule repository, and forms a new source of behavioral constraints for individuals in the two departments.

The university setting exhibits the need for both static and dynamical behavioral constraints, as both single actions (reserving a room) and multi-step transactions (planning a two-semester sequence of courses) are subject to constraints. The regulatory regimes impacting behavior come from multiple sources, and can themselves be dynamic, as policies change and temporary agreements form between subsets of the university. A computational law system in this setting can semantically integrate the university's data along with its diverse, overlapping rulesets, and enable individuals to plan and monitor transactions with respect to these rules. Such capabilities in this domain readily extend to governmental regulatory regimes, and to the business world, especially given its growing reliance of electronically-mediated activity.

4. CURRENT APPROACHES

Electronic commerce, including all business activities conducted on the web (not merely financial transactions), constitutes a ripe domain for computational law. Software companies are beginning to develop tools to assist individuals in imposing rules to structure their electronic activities. Industry consortiums and companies conducting business online have developed workflow and business process languages to help structure electronic transactions, to ensure that they conform to procedural requirements, and are extending these formalisms to mediate interactions with external partners and customers. In the following section, we analyze the capabilities of these formalisms in addressing the demands of computational law.

4.1 Software Tools

Many individual email programs currently offer rule-based filtering systems. These systems, which allow users to construct their own rules using a logic-based language, perform a small portion of the computational law task (regimentation) by enforcing policies for transferring, forwarding, or replying to email messages. However, these tools are not integrated–in general, they cannot incorporate rules from other sources, and do not speak to other programs–and are string-based, not semantic. For example, one can draft a rule that deletes all messages coming from a specific domain, but one cannot draft a rule that deletes all messages from Harvard graduates. However, researchers are encouraging the development of semantically-aware rule-based email systems capable of such expressions[1]. Currently, these systems best contribute to computational law by familiarizing individuals with logic-based interfaces to rules and laws.

4.2 Business Process Languages

Business process languages arose out of specialized systems used by companies to monitor specific business activities, and are closely related to workflow languages and systems meant to control the execution of these activities. One emerging standard business process language is Business Process Execution Language for Web Services (BPEL4WS) [2].

A business process modeled in BPEL4WS specifies a flowchart-like scheme for executing a set of actions, including receiving requests from clients, invoking web services, and composing results to respond to the client (see [18]). A designer of a BPEL4WS process essentially writes a program in this language for the execution of the process, and can enrich the execution of the set of actions in the process using structured constructs (sequence, parallel, loop, case) as well as handle exceptions. A modular composition of actions makes individual changes to the action definitions easy, but since the specification of process execution is a complete program, additional constraints on execution may require remodeling the entire process. Further, the language provides no direct support for constraints on actions that come from business rules, regulations, contracts, or other laws or rulesets: standards for such rules have yet to emerge, although OMG (the Object Management Group) is coordinating industry efforts to this end [11]. Individual software vendors have designed their own rule engines, but without a common rule format, and often with different languages for human-editable rules and machine-processable rules.

4.3 Collaboration Protocol Agreements

Business rule systems have sought to expand beyond monitoring and control of internal processes by additionally representing external business partner interactions and agreements. Expanding on earlier efforts to define a language for specifying trading partner agreements [14], the ebXML project has produced XML-based tools to assist business partners in establishing protocols for electronically-mediated interactions, called Collaboration Protocol Profiles (CPP) and corresponding Collaboration Protocol Agreements [10]. This effort is based on structured templates of agreeable process definitions and terms of service (the CPPs) that can be matched and commonly agreed to, with participants then

conducting their transactions as dictated by agreements (the CPAs) based on the CPPs. With this CPA in place, business partners can execute choreographed business processes. However, on the surface these formalisms are merely definitional, and offer no inherent support for planning or monitoring, nor for integrating with other CPAs or other sources of constraints on business transactions.

4.4 Transaction Logic and CTR-S

Transaction Logic (TR)[3] offers theoretically-grounded behavioral modeling capability that subsumes many of the above, industry-driven formalisms. TR has the ability to model workflows [8] and constraints on business transactions, and its extension, CTR-S, has been specifically applied to web service contracts [6]. Formulas in transaction logic hold over sequences of database states, addressing the transitions taking place over states, not the single states themselves: this gives TR the ability to represent both static as well as dynamic, behavioral constraints. CTR-S, a version of TR enhanced with concurrent (parallel) structures, models workflows that represent a series of transactions between parties, and supports augmenting these workflows with constraints that restrict the possible executions of the workflow. In contrast to a process definition in a business process language, a workflow modeled in CTR-S is not a complete program: the transaction logic description of the workflow admits many possible executions, each of which is a sequence of states reflecting the transactions in the workflow. This representation is significantly more declarative and flexible than traditional business process/workflow representations. CTR-S also offers significant benefits over business process languages by easily integrating additional constraints into the definition of a process or workflow. One limitation of CTR-S is that it models one agent as the reasoner, seeking to augment the workflow with additional constraints, and all other agents in the world as a monolithic opponent– there is no unified view of multiple agents acting simultaneously in an environment.

4.5 Propositional Nets

A new approach to a formalization of computational law uses a logical encoding of propositional nets, a behavioral model that combines features of both state machines and Petri nets. Propositional nets model discrete, synchronous, dynamic systems, support simultaneous actions of multiple agents and offer fine-grained environmental modeling at the level of propositions. Propositional nets are composed of two layers, explicitly representing both the semantic relationships among state data as well as the transitional model of an enterprise that captures the changes to that data based on agent behavior, as well as constraints on those actions, leveraging the semantically rich propositions. The logical encoding of a propositional net supports algorithms for performing the desired operations of a computational law system. Further development of the propositional net model, along the with the specification of these algorithms is the subject of forthcoming work.

5. CONCLUSION AND FUTURE WORK

In this paper, we have argued that reasoning with rules, and the application of logic-based techniques to law, can reach its potential in the domain of electronically-mediated transactions. Examining current techniques for negotiating business rules, regulations, laws, and contracts in this domain, we find a need for a new approach addressing both the special characteristics and the desired operations within electronic commerce, as well as the incomplete, indirect, and overlapping nature of the laws that apply there. We acknowledge that electronically-mediated domains do not bypass all obstacles that have previously confronted rule-based reasoning, and these issues will need to be confronted by future work. For example, efforts to integrate computational law systems with web services will need to address the same issues that web service composition efforts have been concerned with, a problem related to open texture: while web service specifications may refer to concrete terms, the meanings of these terms may not be commonly agreed on by the provider and users of a service. Future work will further explicate the theory of computational law and explore the propositional net formalism, demonstrating its capabilities for assistance in querying, planning, monitoring, regimentation, automated legislation/contract formation, and problem determination, as well as its advantages over existing, implemented approaches in electronically-mediated domains.

6. REFERENCES

[1] J. Alferes, A. Brogi, J. Leite, and L. M. Pereira. An evolvable rule-based e-mail agent. *Lecture Notes in Computer Science*, 2902:394 – 408, Nov. 2003.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Specification: Business process execution language. `http://www-106.ibm.com/developerworks/library/ws-bpel/`, May 2003.

[3] A. J. Bonner and M. Kifer. A logic for programming database transactions. In *Logics for Databases and Information Systems*, pages 117–166. Kluwer Academic Publishers, 1998.

[4] L. K. Branting. Advisory systems for pro se litigants. In *ICAIL '01: Proceedings of the 8th international conference on Artificial intelligence and law*, pages 139–146, New York, NY, USA, 2001. ACM Press.

[5] S. M. Brasil and B. B. Garcia. Modelling legal reasoning in a a mathematical environment through model-theoretic semantics. In *ICAIL '03: Proceedings of the ninth international conference on Artificial intelligence and law*, 2003.

[6] H. Davulcu, M. Kifer, and I. V. Ramakrishnan. CTR-S: A logic for specifying contracts in semantic web services. In *WWW2004 Conference Proceedings*, pages 144–153. Association for Computing Machinery, May 2004.

[7] Intuit, Inc. Turbotax. `http://www.turbotax.com`.

[8] M. Kifer. Transaction logic for the busy workow professional. Technical report, SUNY at Stony Brook, 1996.

[9] J. A. Meldman. A structural model for computer-aided legal analysis. *Rutgers Journal of Computer Law*, 6, 1977.

[10] OASIS ebXML Collaboration Protocol Profile and Agreement Technical Committee. Collaboration protocol profile and agreement specification (version 2.0). `http://www.oasis-open.org/committees/`

download.php/204/ebcpp-2.0.pdf, Sept. 2002.

[11] Object Management Group's Business Enterprise Integration Domain Task Force. Issues request for information on business rule management. http://www.omg.org/technology/technology_adoption/bei-tf.htm, 2004.

[12] E. L. Rissland, K. D. Ashley, and R. P. Loui. AI and law: a fruitful synergy. *Artif. Intell.*, 150(1-2):1–15, 2003.

[13] E. L. Rissland and D. B. Skalak. CABARET: rule interpretation in a hybrid architecture. *Int. J. Man-Mach. Stud.*, 34(6):839–887, 1991.

[14] M. Sachs and J. Ibbotson. Electronic trading-partner agreement for e-commerce (tpaml). http://www.ibm.com/developer/xml/tpaml/tpaspec.pdf, Jan. 2000.

[15] U. J. Schild and S. Herzog. The use of meta-rules in rule based legal computer systems. In *ICAIL '93: Proceedings of the fourth international conference on Artificial intelligence and law*, pages 100–109. ACM Press, 1993.

[16] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The british nationality act as a logic program. *Commun. ACM*, 29(5):370–386, 1986.

[17] A. von der Lieth Gardner. *An artificial intelligence approach to legal reasoning*. MIT Press, 1987.

[18] S. Weerawarana and F. Curbera. Business process with BPEL4WS. http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/, Aug. 2002.