Toward Machine-Understandable Contracts

Sudhir Agarwal[1] and **Kevin Xu** [2] and **John Moghtader** [3]

Abstract. We present Contract Definition Langauge, a novel approach for defining contracts declaratively in a machine-understandable way to achieve better comprehensibility and higher efficiency of reasoning, analysis and execution of contracts through higher degree of automation and interoperability. The effect of representing contracts with our Contract Definition Language is not only a significant reduction of legal transaction costs, but it also opens a variety of new options to create better contracts. As a proof of concept, we also present our modeling of two US statutes (FERPA/SOPIPA/COPAA and HIPAA) as well as our prototypes for validity checking and hypothetical analysis of contracts according to those statues.

1 Introduction

The conventional view is that the automation of contract creation, execution, and compliance is beyond the capabilities of today's technologies. This view stems from a longstanding tradition of contracting practices, where all terms and conditions are expressed and interpreted in natural language, often in obtuse, inaccessible legalese that can only be deciphered by lawyers and judges. Many legal scholars have called for more usable, interactive tools to make better sense of contracts [1, 4, 8, 11, 14]. Contract scholars have defined a need for user-facing tools that would make contracts more understandable and actionable [4, 12], as well as a more modular and machine-readable approach to future contracting practices [3, 15, 16].

The impetus for the Computable Contracts research at Stanford [4], and Computational Law [5] more broadly, is a vision where computers, or people with the help of computers, are able to rapidly understand the implications of contracts in their legal context, in order to make optimal decisions accordingly, on a potentially large and complex scale. This vision, if realized, will dramatically improve access to justice in the legal system.

In our definition, computable contracts have the following main features:

1. **Machine-understandable**: In contrast to traditional natural language contracts, computable contracts have logic-based formal semantics which enables use of machines to automatically reason over contracts.
2. **Declarative**: In contract to hard-coding contract terms with a procedural programming language such as Java and C++, computable contracts are defined declaratively. Thus computable contracts can be better comprehensible by legal professionals as well as the clients as the declarative nature is closer to the way domain knowledge is specified.
3. **Executable**: Computable contracts are executable like procedural code. Thus, computable contracts do not need to be translated to programmed in a traditional programming language which reduces costs and errors as there is not need to manage two separate versions (one for humans, another for machines) of the same contract.
4. **Interoperable**: Computable contracts are interoperable in the sense that they use shared vocabulary for referring to real world objects, thus enabling automating reasoning over multiple different contracts that may have interdependencies or even conflicting terms.

In this paper, we present an approach for formulating, analyzing, and executing contracts more efficiently and effectively by enabling a high degree of automation. We introduce a Logic Programming based Contract Definition Language (CDL). CDL makes possible automated reasoning over legal substance. We have identified the following four types of reasoning tasks with contracts:

1. **Validity Check**: Validity checking determines whether a contract satisfies all the constraints it must satisfy. For example, a marriage contract between an adult and a minor is invalid in most countries because the law of the countries does not allow such contracts. Constraints that need to be satisfied can be defined with CDL as well.
2. **Hypothetical Analysis**: In many cases, a user (e.g. a legal professional, a client, a customer, an employee etc.) wishes to understand a given contract or a set of contracts for a situation. For example, an employee who is not keeping very good health may be interested in knowing what happens when he/she has to take more sick leave than mentioned in the contract. Hypothetical analysis roughly provides an answer to the question: What are the implications (obligations/rights) of laws and/or contract terms in a particular, given or hypothetical, situation?
3. **Utility Computation**: Terms in contracts often have a different utility for the involved parties. The utility depends on the party's preferences. When a party's preferences are known, the utility of a contract for the party can be computed automatically.
4. **Planning**: A contract can be seen as a set of constraints on involved parties' behavior. When the goal of a party is known, a plan, i.e. a sequence of actions, can be computed automatically such that the execution of the sequence of actions would lead the party to its desired goal state. Planning problem has been extensively studied as a discipline of Computer Science and Artificial Intelligence [5], and it might be possible to adopt one of the existing techniques for our purpose.

As far as reasoning with contracts is concerned, in this paper, we focus on validity check and hypothetical analysis only. They allow

[1] CS Dept., Stanford University, USA. Email: sudhir@cs.stanford.edu
[2] Law School, Stanford University, USA. Email: kevin.s.xu@gmail.com
[3] Law School, Stanford University, USA. Email: jmoghtader@gmail.com
[4] `http://compk.stanford.edu`
[5] `http://complaw.stanford.edu`

automated support while still leaving the decision-making and implementation to the user, and are also required by other reasoning tasks.

The paper is organized as follows. We first give an overview of foundations upon which our technique is built. Then, we present the syntax and semantics of our Contract Definition Language (CDL). We present two case studies involving modeling with CDL and automatically reasoning about two U.S. Federal statutes, FERPA and HIPAA. We conclude and identify next steps after a discussion of related work.

2 Foundations

In this section we give a short overview of the syntax and intuitive semantics of deductive databases and logic programs, two foundational techniques upon which we build our Contract Definition Language.

2.1 Databases

The *vocabulary* of a database is a collection of object constants, function constants, and relation constants. Each function constant and relation constant has an associated arity, i.e. the number of objects involved in any instance of the corresponding function or relation. A term is either a symbol or a functional term. A functional term is an expression consisting of an n-ary function constant and n terms. In what follows, we write functional terms in traditional mathematical notation - the function followed by its arguments enclosed in parentheses and separated by commas. For example, if `f` is a binary function constant and if `a` and `b` are object constants, then `f(a,a)` and `f(a,b)` and `f(b,a)` and `f(b,b)` are all functional terms. Functional terms can be nested within other functional terms. For example, if `f(a,b)` is a functional term, then so is `f(f(a,b),b)`. A datum is an expression formed from an n-ary relation constant and n terms. We write data in mathematical notation. For example, we might write `parent(art,bob)` to express the fact that Art is the parent of Bob. A dataset is any set of data that can be formed from the vocabulary of a database. Intuitively, we can think of the data in a dataset as the facts that we believe to be true in the world; data that are not in the dataset are assumed to be false.

2.2 Logic Programs

The language of logic programs includes the language of databases but provides additional expressive features. One key difference is the inclusion of a new type of symbol, called a *variable*. Variables allow us to state relationships among objects without explicitly naming those objects. In what follows, we use individual capital letters as variables, e.g. `X`, `Y`, `Z`. In the context of logic programs, a term is defined as an object constant, a variable, or a functional term, i.e. an expression consisting of an n-ary function constant and n simpler terms. An atom in a logic program is analogous to a datum in a database except that the constituent terms may include variables. A *literal* is either an atom or a negation of an atom (i.e. an expression stating that the atom is false). A simple atom is called a *positive* literal, The negation of an atom is called a *negative* literal. In what follows, we write negative literals using the negation sign `~`. For example, if `p(a,b)` is an atom, then `~p(a,b)` denotes the negation of this atom. A *rule* is an expression consisting of a distinguished atom, called the head and a conjunction of zero or more literals, called the *body*. The literals in the body are called *subgoals*. In what follows, we write rules as in the example `r(X,Y) :- p(X,Y) & ~q(Y)`. Here, `r(X,Y)` is the head, `p(X,Y) & ~q(Y)` is the body; and `p(X,Y)` and `~q(Y)` are subgoals.

Semantically, a rule states that the conclusion of the rule is true whenever the conditions are true. For example, the rule above states that `r` is true of any object `X` and any object `Y` if p is true of `X` and `Y` and `q` is not true of `Y`. For example, if we know `p(a,b)` and we know that `q(b)` is false, then, using this rule, we can conclude that `r(a,b)` must be true.

3 Contract Definition Language (CDL)

CDL descriptions are open logic programs. While a traditional logic program is typically used to specify views and constraints on a single database state, CDL descriptions can specify a state-transition system. CDL is expressive enough to define a Turing machine. The declarative syntax and formal semantics of CDL makes CDL descriptions easier to comprehend and maintain. The executability of CDL descriptions makes it superfluous to hard-code contract terms with procedural code as well as makes CDL a promising alternative for defining self-executable contracts such as Ethereum Smart Contracts [6, 10].

The basis for CDL is a conceptualization of contracts in terms of entities, actions, propositions, and parties. Entities include objects relevant to the state of a contract are usually represented by object constants in CDL. In some cases, we use compound terms to refer to entities. Actions are performed by the parties involved in the contract. As with entities, we use object constants or compound terms to refer to primitive actions. Some actions may not be legal in every state. Propositions are conditions that are either true or false in each state of a contract. In CDL, we designate propositions using object constants or compound terms. Parties are the active entities in contracts. Note that, in each state, some of the contract's propositions can be true while others can be false. As actions are performed, some propositions become true and others become false. On each time step, each party has a set of legal actions it can perform and executes some action in this set. In CDL, we usually use object constants (in rare cases compound terms) to refer to parties. In CDL, the meaning of some words in the language is fixed for all contracts (the contract-independent vocabulary) while the meanings of all other words can change from one contract to another (the contract-specific vocabulary).

There are the following contract-independent structural relation constants.

1. `role(r)` means that r is a role in the contract.
2. `base(p)` means that `p` is a base proposition in the contract.
3. `percept(r,p)` means that `p` is a percept for role `r`.
4. `input(r,a)` means that `a` is an action for role `r`.

To these basic structural relations, we add the following relations for talking about steps.

1. `step(s)` means that `s` is a step.
2. `successor(s1,s2)` means that step `s1` comes immediately before step `s2`.
3. `true(p,s)` means that the proposition `p` is true on step `s`.
4. `sees(r,p,s)` means that role `r` sees percept `p` on step `s`.
5. `does(r,a,s)` means that role `r` performs action `a` on step `s`.
6. `legal(r,a,s)` means it is legal for role `r` to play action `a` on step `s`.

7. `goal(r,n,s)` means that player has utility `n` for player `r` on step `s`.
8. `terminal(s)` means that the state on step `s` is terminal.

The truth of propositions in the initial state can be stated using true with the first step as the step argument; and update rules can be stated using `true` and `successor`. Just how this works should become clear from the following modeling of a part of the well known board game Tic-Tac-Toe.

There are two roles say `black` and `white` who make their moves alternatively.

```
role(white)
role(black)
true(control(black),N) :-
    true(control(white),M) &
    successor(M,N)
true(control(black),N) :-
    true(control(white),M) &
    successor(M,N)
```

If `white` marks a cell in a state, then that cell has a `x` in the next state. Analogously, if `black` marks a cell in a state, then that cell has a `o` in the next state. Further rules not shown below ensure that all other cells carry their previous markings to the next state.

```
true(cell(I,J,x),N) :-
    does(white,mark(I,J),M) &
    successor(M,N)
true(cell(I,J,o),N) :-
    does(black,mark(I,J),M) &
    successor(M,N)
```

CDL interprets negations as failure and does not allow negations or disjunctions in the head. While negation as failure could be a limitation in some scenarios, in many scenarios one can safely make a closed-world assumption. The inability to express disjunctions in the head can be a limitation in some cases, but in many cases, the regulations are laws and regulations are definite. In many cases, once can introduce a new atom to represent the union of the disjunts. Exceptions can be modeled with CDL indirectly by introducing an auxiliary view and adding its negation in the body of the appropriate rules.

4 Case Studies

Thus far, we have modeled two sets of U.S. Federal statutes: 1. the intersecting compliance requirements of the Family Educational Rights and Privacy Act, Children's Online Privacy Protection Act, and Student Online Personal Information Protection Act (FERPA/COPPA/SOPIPA); 2. the Privacy Rule in Health Insurance Portability and Accountability Act (HIPAA). Both sets of statutes are complex in their content and form. Our work in modeling these statutes is motivated by the goal of demonstrating how a computable contract scheme can increase accuracy, efficiency, and consistency in the interpretation of complex legal landscapes, which would be valuable for both lawyers working in those domains and laypeople who are affected by the relevant laws.

4.1 FERPA Prototype

In this prototype, we have modeled with CDL the intersecting compliance requirements of the Family Educational Rights and Privacy Act, Children's Online Privacy Protection Act, and Student Online Personal Information Protection Act (FERPA/COPPA/SOPIPA). The prototype is online available at `http://compk.stanford.edu/ferpa.html`.

The prototype allows interactive formation of an agreement between an information service provider and a district as well as analysis of multiple agreements. It demonstrates the capabilities and added value of our proposed language and reasoning techniques in the situation where an information service provider wishes to obtain data about school going children. In such a case, the information service provider is required by law to enter into a contract with the districts controlling the schools. In the prototype, a contacting party can fill in the details such as which student's data is to be shared, the potential use of the data by the provider, age/grades level of the students etc. The prototype then checks the validity of the contract as per FERPA, COPPA and SOPIPA and displays the violations and obligations if any. The behavior of the user interface is directly determined by FERPA, COPPA and SOPIPA rules modeled with CDL. For the purpose of the demo, we allow users to edit the rules and verify the change in the behavior of the system.

Below we present an excerpt of the database and the rules that we have modeled for this case study. The complete set of rules is visible in the 'Law' tab of the prototype.

4.1.1 Views

The following view definitions define the categories `district_data_pii`. Categories `district_data_non_pii`, `additional_data_pii` and `additional_data_non_pii` can be defined analogously.

```
district_data_pii(D,district_student_name) :-
   district_data(D,district_student_name)

district_data_pii(D,district_student_
   parent_name) :-
   district_data(D,district_student_parent_name)

district_data_pii(D,district_student_dob) :-
   district_data(D,district_student_dob)

district_data_pii(D,district_student_address):-
   district_data(D,district_student_address)

district_data_pii(D,district_student_ssn):-
   district_data(D,district_student_ssn)

district_data_pii(D,district_student_moms_
   maiden_name):-
   district_data(D,district_student_moms_
   maiden_name)

district_data_pii(D,district_student_pob):-
   district_data(D,district_student_pob)
```

The following view definition states a provider is under direct control of district if the provider can amend terms with consent. Other views can be modeled analogously.

```
provider_under_direct_control_of_district(D) :-
   provider_can_amend_terms_with_consent(D)
```

For the complete set of view definition we refer to the ‘Law’ tab of the prototype.

4.1.2 *Constraints*

Below the CDL modeling of the constraint that for any district data PII, if the selected FERPA provision is school official exemption, then the provider must be under direct control of the district. Other constraints can be modeled analogously.

```
illegal("Provider must be under direct control
   of District.") :-
   district_data_pii(D,A) &
   ferpa_provision(D,school_official_exemption) &
   ~provider_under_direct_control_of_district(D)
```

The following rule states the provider must select a FERPA provision.

```
illegal("Must select FERPA exemption.") :-
   district_data_pii(D,A) &
   ~ferpa_provision(D,actual_parental_consent) &
   ~ferpa_provision(D,directory_exemption) &
   ~ferpa_provision(D,school_official_exemption)
```

The following rule states that commercial use of data is prohibited under school official exemption.

```
illegal("Under School Official Exemption,
   commercial use of data is prohibited."):-
   ferpa_provision(D,school_official_exemption) &
   district_potential_use_by_provider(
   D,district_4aiii)
```

The following rule states that if the district is in California, then commercial use of any data from the district is prohibited.

```
illegal("SOPIPA prohibits commercial use of
   any data.") :
   district_in_california(D) &
   district_data_pii(D,A) &
   district_potential_use_by_
   provider(D,district_4aiii)
```

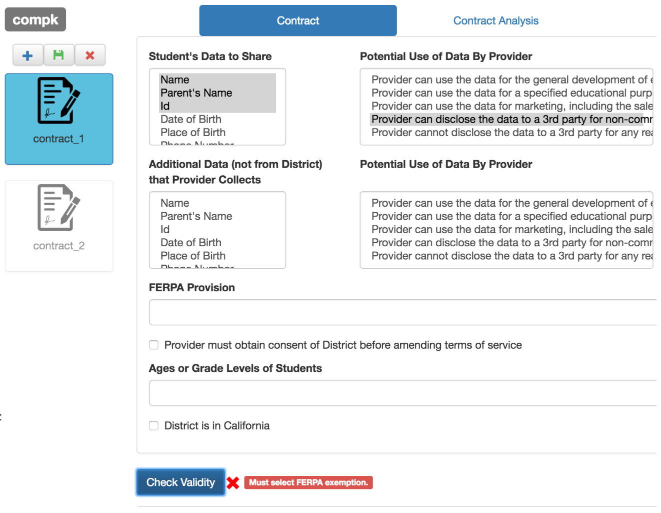
For the complete set of modeled view definition we refer to the ‘Law’ tab of the prototype.

4.1.3 *Obligations*

Below the CDL modeling of the consequence that if FERPA provision is directory exemption, then the district must allow opportunity for parent to opt-out of the disclosure of student’s data.

```
add(D,"District must allow opportunity for
   parents to opt-out of the disclosure of
   student+data.") :-
   district_data_pii(D,A) &
   ferpa_provision(D,directory_exemption)
```

For the complete set of modeled obligations we refer to the ‘Law’ tab of the prototype.



Student's Data to Share

Name
Parent's Name
Id
Date of Birth
Place of Birth

Potential Use of Data By Provider

Provider can use the data for the general development of
Provider can use the data for a specified educational purp
Provider can use the data for marketing, including the sale
Provider can disclose the data to a 3rd party for non-comr
Provider cannot disclose the data to a 3rd party for any rea

Additional Data (not from District) that Provider Collects

Name
Parent's Name
Id
Date of Birth
Place of Birth

Potential Use of Data By Provider

Provider can use the data for the general development of
Provider can use the data for a specified educational purp
Provider can use the data for marketing, including the sale
Provider can disclose the data to a 3rd party for non-comr
Provider cannot disclose the data to a 3rd party for any rea

FERPA Provision

☐ Provider must obtain consent of District before amending terms of service

Ages or Grade Levels of Students

☐ District is in California

Must select FERPA exemption.

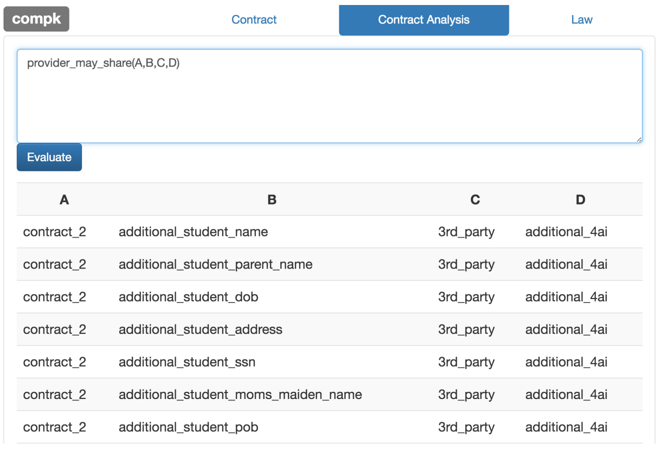
Figure 1. Screenshot of our prototype showing that the contract is invalid as well as the reason for the invalidity.

Validity Checking As briefly mentioned above, our prototype can automatically check whether a contract is valid according to a law. This feature can be very useful in the contract formation phase. Figure 1 shows an example contract in which the information service provider is not under the control of the district, and therefore the selected data artifacts to be shared for the selected intended use is not allowed by the law. In the example contract shown in Figure 1, the violation is computed because of the presence of the following rule.

```
illegal("Provider must be under direct control
   of District.") :- district_data_pii(D,A) &
   ferpa_provision(D,school_official_exemption) &
   ~provider_under_direct_control_of_district(D)
```

In addition to computing whether a contract is valid or not, our prototype can also automatically output the reason for the invalidity or any rights, limitations and obligations that parties have if the contract is valid. In our example, if the FERPA is changed to ”Actual Parental Consent”, then the contract becomes valid, and our prototype can also automatically compute that the district may disclose the data only according to the terms of the parental consent.

Hypothetical Analysis In addition to validity checking as described above, the FERPA prototype also supports hypothetical reasoning over a set of (valid) contracts. Typically, an information service provider enters into multiple contracts, one for each district, to be able to achieve broader coverage for his/her service. Given a set of such contracts, an information provider is often faced with the problem of deciding whether he/she may use certain data artifact for a particular use. In order to obtain the answer to such a question with our prototype, the information service provider would formulate his question as a query in the ‘Contract Analysis’ tab. Our prototype then analyses all the existing contracts of the information service provider and produces the answer to the question. For example, if an information service provider wishes to know which data he/she may share with a third party, he/she would pose the query `provider_may_share(District,Data,3rd_party,Use)`. The answer to this query will contain all (district, data artifact, usage) that the provider may share with a 3rd party (see also Figure 2).



provider_may_share(A,B,C,D)

A	B	C	D
contract_2	additional_student_name	3rd_party	additional_4ai
contract_2	additional_student_parent_name	3rd_party	additional_4ai
contract_2	additional_student_dob	3rd_party	additional_4ai
contract_2	additional_student_address	3rd_party	additional_4ai
contract_2	additional_student_ssn	3rd_party	additional_4ai
contract_2	additional_student_moms_maiden_name	3rd_party	additional_4ai
contract_2	additional_student_pob	3rd_party	additional_4ai

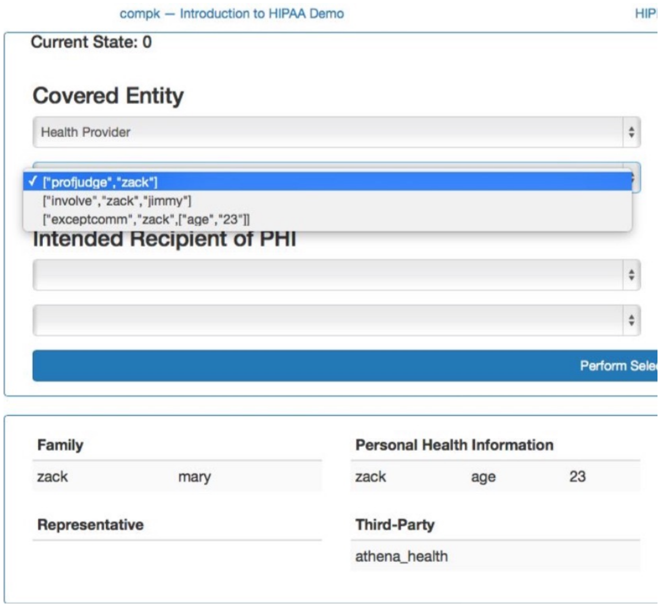
Figure 2. Screenshot of the output of hypothetical analysis

4.2 HIPAA Prototype

In our HIPAA prototype, we specifically modeled the Privacy Rule of the statute. Under the HIPAA Privacy Rule, all types of healthcare providers and related organizations, collectively known as ‘covered entities,’ are the main subjects of regulation and must comply with the Rule’s provisions when deciding if and how to disclose a patient’s information, called Protected Health Information (PHI). The Rule’s complexity can hinder compliance and lead to potentially huge penalties for parties that violate the Rule. Thus, providing clarity using a computational contract scheme can be immediately valuable to all parties involved. Because the core of HIPAA privacy compliance is what covered entities can and cannot do with a patient?s PHI, we modeled a couple of situations that resemble hypothetical analysis and validity checking, where using a computational contract scheme can help covered entities navigate their legal options. The prototype is available at `http://compk.stanford.edu/hipaa.html`.

Scenario 1 Covered Entity, X, wants to disclose patient Y’s information to third party Z, who is a healthcare startup that wants to market its products to Y. Under HIPAA, such disclosure is only legal if Y provides explicit written authorization in plain language for X to issue such disclosure. Any type of non-written authorization would not be valid. In this situation, in order for the covered entity to comply with this constraint, our prototype provides only the option of disclosing PHI for marketing purposes available for X, if a written authorization is issued by Y, allowing X to analyze what are its legal courses of action. This dynamic is modeled by the following rules:

```
legal (X, market(Phi, Z), N) :-
   true(written_authorization(Y, X), M) &
   true(plain_language(Y, X), M) &
   phi (Y, Phi) & ce(X) &
   thirdparty(X, Z) &
   successor (M, N)

true (plain_language(Y, X), M) :-
   does (Y, write_plain_lang(X), M)

true (plain_language(Y, X), N) :-
   true (plain_language(Y, X), M) &
   successor (M, N)

true (written_authorization(Y, X), M) :-
does (Pa, write_authorization(X), M)

true (written_authorization(Y, X), N) :-
   true (written_authorization(Y, X), M) &
   successor(M, N)

legal (X, market(Phi, Y), M) :-
   step(M) &
   does (X, exceptcomm(Y, exception), M) &
   ce(X) & phi(Y, Phi) & successor
```


compk — Introduction to HIPAA Demo
HIP
Covered Entity
Intended Recipient of PHI
Perform Sele
Family
Personal Health Information
Representative
Third-Party

Current State: 0

Health Provider

✓ ["profjudge","zack"]
["involve","zack","jimmy"]
["exceptcomm","zack",["age","23"]]

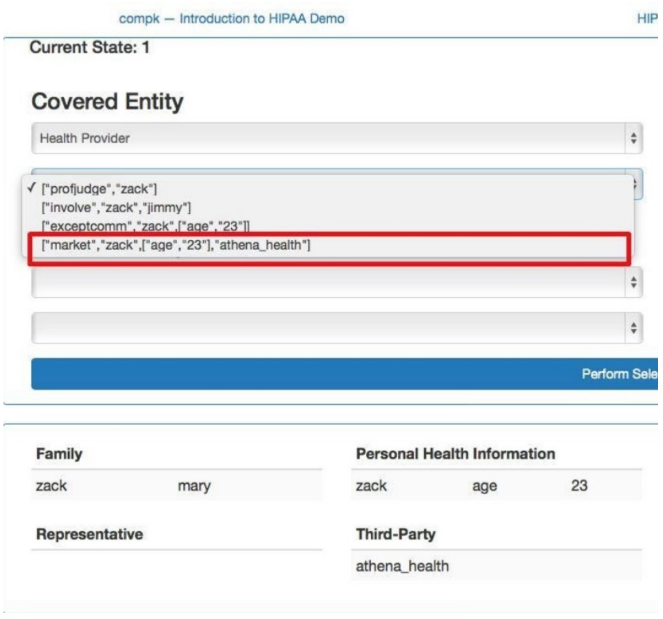
Family		Personal Health Information		
zack	mary	zack	age	23

Representative	Third-Party
	athena_health

Figure 3. Screenshot showing a health provider’s options when it does not have a written permission of a patient

Figure 3 illustrates the scenario, in which the three actors are Zack (the patient), Kantor (Zack’s health provider), and Athena (a third party, healthtech startup). Kantor wants to disclose Zack’s age, which is a type of PHI specified in HIPAA, to Athena for marketing purposes. In order for Kantor to do this legally, Zack must provide written authorization to Kantor, permitting this disclosure. (Note: non-written permission would not work.) Thus, in state 0, the beginning, when Kantor has not obtained Zack’s written permission, Kantor?s options do not include ‘marketing’.

However, if Zack does provide written authorization in state 0 (in our product, you would select Zack in the patient box, select the ”writtenauth” option under Zack, then click ”Perform Selected Actions”), it will move things to state 1, the next state, and Kantor will have the marketing option available to disclose to a third party, like Athena (see Figure 4)


compk — Introduction to HIPAA Demo

HIP

Current State: 1

Covered Entity

Health Provider

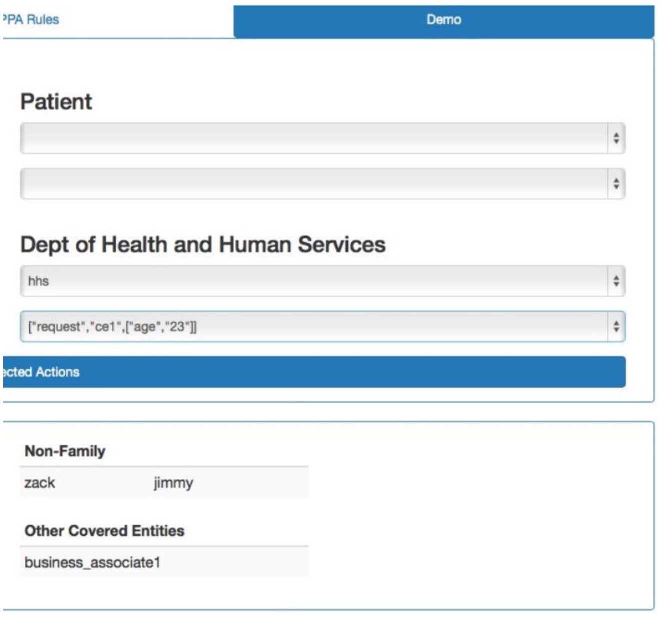
✓ ["profjudge","zack"]
["involve","zack","jimmy"]
["exceptcomm","zack",["age","23"]]
["market","zack",["age","23"],"athena_health"]

Perform Sele

Family		Personal Health Information		
zack	mary	zack	age	23

Representative	Third-Party
	athena_health

Figure 4. Screenshot showing a health provider's options when it has a written permission of a patient


PA Rules

Demo

Patient

Dept of Health and Human Services

hhs

["request","ce1",["age","23"]]

cted Actions

Non-Family	
zack	jimmy

Other Covered Entities
business_associate1

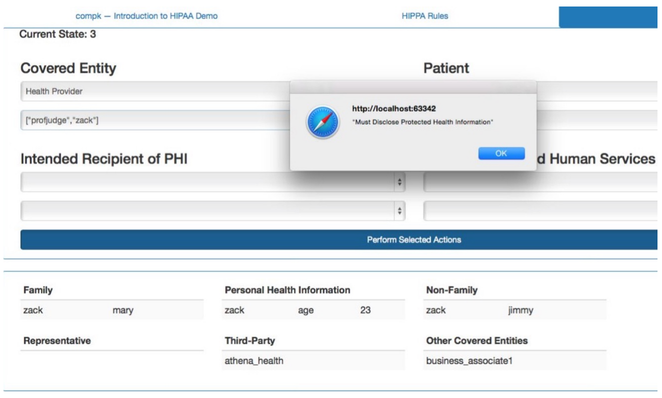
Figure 5. Screenshot showing HHS requesting a health provider to disclose a patient's age.

Scenario 2 this situation models an interaction between Covered Entity, X, and the U.S. Department of Health and Human Services (HHS), which is the main federal agency that regulates and enforces HIPAA compliance. Under the law, if HHS requests PHI from a covered entity, regarding say patient Y, for the purpose of investigating a compliance case, the covered entity must disclose the information to HHS. To depict this constraint in our demo, as soon as HHS issues a request for PHI, X's options are limited to just one: disclose. In fact, if X chooses any other option, our system will conduct validity checking and not allow the step to proceed. Instead, it will immediately alert X that it must disclose the requested PHI. This dynamic is modeled by the following rules:

```
illegal ("Must Disclose PHI") :-
   true(requestment(HHS, X, Phi), M) &
   true(investigation(HHS, X), M) & ce(X) &
   ~does (X, disclose (Phi, HHS), N) &
   phi(Y, Phi) & successor (M, N)

illegal ("Must Disclose PHI")  :-
   does(Y, request(X, Phi), M) & ce(X) &
   phi (Y, Phi) & successor (M, N)&
   ~does (X, disclose (Phi, HHS), N)

illegal ("Must Disclose PHI")  :-
   does (Rep, request(X, Phi), M) &
   phi (Y, Phi) & ce(X) &
   rep (Y, Rep) & successor (M, N) &
   ~does (X, disclose (Phi, HHS), N)
```

This scenario illustrated in Figure 5 is between Kantor and HHS. In HIPAA, when HHS requests PHI from a covered entity like Kantor, for example, if HHS received complaints by patients regarding potential HIPAA violations and wanted to investigate Kantor, Kantor is required to disclose the requested information to HHS, which limits Kantor's legal options. Here in state 2, HHS is requesting Kantor to disclose Zack's age as a part of its investigation.

After HHS commits this request, we move to state 3, where Kantor is legally obligated to disclose Zack's age to HHS. If Kantor does not disclose and instead chooses to commit another action, it is considered illegal; therefore, our system will show an alert stating that PHI must be disclosed and will not move on to the next state until Kantor commits the correct, legally required action (see Figure 6).


compk — Introduction to HIPAA Demo

HIPPA Rules

Current State: 3

Covered Entity

Patient

Health Provider

["profjudge","zack"]

http://localhost:63342
"Must Disclose Protected Health Information"

OK

Intended Recipient of PHI

d Human Services

Perform Selected Actions

Family		Personal Health Information			Non-Family	
zack	mary	zack	age	23	zack	jimmy

Representative	Third-Party	Other Covered Entities
	athena_health	business_associate1

Figure 6. Screenshot showing the enforcement of a health provider's obligation to disclose a PHI of a patient.

5 Related Work

One of the first approaches for formalizing regulations with Logic Programs was presented in [13]. One of the major differences between CDL and the approach presented in [13] is that with CDL one can also express a multi-actor dynamic system and constraints over multiple states.

The formalism presented in [2] can reason over data types (but not over data values) of single individuals (but not of groups), and cannot express certain norms precisely due to lack of parameterized roles. A formalism called pLogic presented in [7] does not allow reasoning over history and future possibilities of actions of various actors. In contrast, CDL can reason over types and (complex logical) interdependencies of objects. CDL can also express and reason over past states and possible actions of an actor in a state in the future. Our formalization technique is also more general than the one presented in [3] as the latter can express only finite state automata. The formalisms presented in [**?**] can describe contracts in terms of deontic concepts such as obligations, permissions and prohibitions. CDL does not directly have special constructs for deontic modalities. However, since deontic modalities can be reduced to temporal modalities and CDL can express dynamic constraints, it is possible to express deontic modalities with CDL.

The term Computable Contracts has been used with various different meanings in the past. In some cases it refers to computer-aided contract formation tools to support a legal professional in drafting a contract in natural language. Such softwares range from simple extensions to popular text processing systems to collaborative web-based contract drafting such as Beagle [6], ClauseMatch [7], and Nitro [8] or more efficient contract readers such LegalSifter [9]. In some other cases such as Kira [10] and LegalRobot [11], natural language contracts are analyzed by extracting terms from contracts. Such analysis techniques are statistics based and cannot do reasoning over complex logical interrelationships of artifacts appearing in a contract. Since such a reasoning capability is a prerequisite for reasoning over dynamics described in a contract, such techniques can also not reason over dynamic constraints on the behavior of the contracting parties.

Recently, the so-called Smart Contracts in Blockchain 2.0, e.g. Ethereum [6, 10], have received a lot of attention [9]. An Ethereum Blockchain is essentially a state-transition system that is managed without central control and secured with cryptographic techniques. Currently, Ethereum Smart Contracts are mostly programmed in a procedural language called Solidity, and thus suffer from the already mentioned problems of procedural hard-coding of contracts. CDL on the other hand is declarative as well as allow for usage of shared vocabulary (e.g. a domain ontology) in multiple contracts. Since the execution semantics of CDL is based on a mapping to state-transition system, we believe that CDL lends itself as a promising alternative to Solidity.

6 Conclusion and Outlook

Our FERPA and HIPAA prototypes have demonstrated two core strengths of making law computable: consistent accuracy and ease of use. Instead of relying on lawyers, whose knowledge on specific subject matters may vary, a service backed up by computable contracts encoded with accurate information can consistently lay out the available legal options for organizations who need help planning their actions, consequently increasing access to justice in particular regulatory areas. This service is also easy to use, for both laypeople and lawyers, because the options being laid out are essentially translated from legalese to plain language that people can understand without prior training. If this type of computable contracts services were to expand, it would not only be valuable to organizations who operate in highly regulated industries, like the covered entities in HIPAA, but also regulatory agencies of these industries, who are in charge of sifting through hundreds of thousands of compliance complaints. For example, from 2003-2013, the number of HIPAA violation complaints sent to HHS increased by more than 900%, and approximately 80% of these complaints are illegitimate, due to simple definitional reasons, e.g. whether the alleged organization falls under the category of covered entities. An extended version of our HIPAA prototype can provide immediate and significant efficiency for HHS, or similar agencies in other countries, to process these types of faulty claims.

One limitation we do acknowledge is our formalization technique's inability to capture inherent nuances that exists in complex legal frameworks. While we assume in our prototypes that every concept has static legal meaning, in reality, many concepts are open to interpretation. This ambiguity often exists by design, because a good piece of law must both incorporate issues of the present day and remain relevant with new behaviors that arise in the future. It is this ambiguity in law that gives rise to disputes and litigations, requiring the input of experienced lawyers and policymakers, who could better anticipate how certain ambiguities would be treated by legislators or judges who have the power to decide what they should mean. This challenge is not something a computable contract scheme is suited to solve, though we can easily envision a future where computable contracts can provide a solid baseline understanding that can be complementary to the work of experienced lawyers, regulators, and lawmakers.

Another area that we will continue to explore is the intersection between computable contracts and the field of Human-Computer Interactions (HCI). With the right kind of user-centric, front-end design, the strength of the expert systems encoded in a computable contract scheme can be easily accessed by ordinary people who need the information contained in these systems, thus significantly enhancing the usability of computable contracts. A complementary partnership between a sleek, intuitive front-end design powered by HCI-principles, and a robust, adaptive back-end system powered by computable contracts could unlock unimaginable benefits for improving access to justice in all societies.

In another thread, we plan to implement a compiler to translate our declaratively specified computable contracts to the EtherScript, the assembly language of Ethereum Virtual Machine while preserving the execution semantics of the contracts. This would enable domain experts and other legal professionals to draft and execute their Smart Contracts themselves instead of getting them programmed by a software developer who may not be a domain expert.

[6] http://beagle.ai
[7] http://clausematch.com
[8] https://www.gonitro.com
[9] https://www.legalsifter.com
[10] https://kirasystems.com/
[11] https://www.legalrobot.com

REFERENCES

[1] K. Adams. Bringing innovation to the tradition of contract drafting: An interview with ken adams. http://blog.scholasticahq.com/post/98149002813/bringing-innovation-to-the-tradition-of-contract, 2014.

[2] Datta A. Mitchell J.C. Barth, A. and H. Nissenbaum. Privacy and contextual integrity: Framework and applications, 2006.

[3] M. D. Flood and O.R. Goodenough. Contract as automaton: The computational representation of financial agreements, 2015.

[4] E. Gerding, 'Contract as pattern language', *Washington Law Review*, **88**(1223), (2013).

[5] Malik Ghallab, Dana S. Nau, and Paolo Traverso, *Automated planning - theory and practice*, Elsevier, 2004.

[6] Evans Jon. Vapor no more: Ethereum has launched. `http://techcrunch.com/2015/08/01/vapor-no-more-ethereum-has-launched/` (retrieved May 2016),.

[7] Mitchell J.C. Lam, P.E. and S Sundaram, 'A formalization of hipaa for a medical messaging system', in *In Proc. of the 6th International Conference on Trust, Privacy and Security in Digital Business (TRUSTBUS 2009)*, (2009).

[8] A.V. Lisachenko, 'Law as a programming language', *Review of Central and East European Law*, **37**, 115–124, (2012).

[9] Swan M., *Blockchain: Blueprint for a New Economy 1st Edition*, O?Reilly Media, 2015.

[10] Arvind Narayanan, Bonneau Joseph, Felten Edward, Miller Andrew, and Goldfeder Steven. Bitcoin and cryptocurrency technologies. `https://d28rh4a8wq0iu5.cloudfront.net/bitcointech/readings/princeton_bitcoin_book.pdf?a=1` (Retrieved May 2016).

[11] Haapio H. Passera, S. and T. Barton. Innovating contract practices: Merging contract design with information design, 2013.

[12] S. Peppet, 'Freedom of contract in augmented reality: The case of consumer contracts', *UCLA Law Review*, **59**(676), (2012).

[13] 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, (May 1986).

[14] H. Smith, 'Modularity in contracts: Boilerplate and information flow', *Michigan Law Review*, **104**(5), 1175–1222, (2006).

[15] H. Surden, 'Computable contracts', *UC Davis Law Review*, **46**(629), (2012).

[16] G.G Triantis, 'Improving contract quality: Modularity, technology, and innovation in contract design', *Stanford Journal of Law, Business, and Finance*, **18**(2), (2013).