University of Pennsylvania
ScholarlyCommons

Departmental Papers (CIS) | Department of Computer & Information Science

7-1-2008

Reasoning about Conditions and Exceptions to Laws in Regulatory Conformance Checking

Nikhil Dinesh
University of Pennsylvania, nikhild@seas.upenn.edu

Aravind K. Joshi
University of pennsylvania, joshi@seas.upenn.edu

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Oleg Sokolsky
University of Pennsylvania, sokolsky@cis.upenn.edu

Postprint version. Presented at the *Ninth International Conference on Deontic Logic in Computer Science, (DEON'08)*, July, 2008.

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/cis_papers/371
For more information, please contact repository@pobox.upenn.edu.

Reasoning about Conditions and Exceptions to Laws in Regulatory Conformance Checking

Abstract
This paper considers the problem of checking whether an organization conforms to a body of regulation. Conformance is cast as a trace checking question – the regulation is represented in a logic that is evaluated against an abstract trace or run representing the operations of an organization. We focus on a problem in designing a logic to represent regulation.

A common phenomenon in regulatory texts is for sentences to refer to others for conditions or exceptions. We motivate the need for a formal representation of regulation to accommodate such references between statements. We then extend linear temporal logic to allow statements to refer to others. The semantics of the resulting logic is defined via a combination of techniques from Reiter's default logic and Kripke's theory of truth.

Comments
Postprint version. Presented at the *Ninth International Conference on Deontic Logic in Computer Science, (DEON'08)*, July, 2008.

This conference paper is available at ScholarlyCommons: http://repository.upenn.edu/cis_papers/371

Reasoning about Conditions and Exceptions to Laws in Regulatory Conformance Checking*

Nikhil Dinesh, Aravind Joshi, Insup Lee, and Oleg Sokolsky

Department of Computer Science
University of Pennsylvania
Philadelphia, PA 19104-6389, USA
{nikhild,joshi,lee,sokolsky}@seas.upenn.edu

Abstract. This paper considers the problem of checking whether an organization conforms to a body of regulation. Conformance is cast as a trace checking question – the regulation is represented in a logic that is evaluated against an abstract trace or run representing the operations of an organization. We focus on a problem in designing a logic to represent regulation.
A common phenomenon in regulatory texts is for sentences to refer to others for conditions or exceptions. We motivate the need for a formal representation of regulation to accomodate such references between statements. We then extend linear temporal logic to allow statements to refer to others. The semantics of the resulting logic is defined via a combination of techniques from Reiter's default logic and Kripke's theory of truth.

1 Introduction

Regulations, laws, and policies that affect many aspects of our lives are represented predominantly as documents in natural language. For example, the Food and Drug Administration's Code of Federal Regulations [1] (FDA CFR) governs the operations of American bloodbanks. The CFR is framed by experts in the field of medicine, and regulates the tests that need to be performed on donations of blood before they are used. In such safety-critical scenarios, it is desirable to assess formally whether an organization (bloodbank) conforms to the regulation (CFR).

There is a growing interest in using formal methods to assist organizations in complying with regulation [2–4]. Assisting an organization in compliance involves a number of tasks related to the notion of a violation. For example, it is of interest to detect or prevent violations, assign blame, and if possible, recover from violations. In this paper, we focus on *conformance checking* which involves detecting the presence of violations.

We cast conformance checking as a trace-checking question. The regulation is translated to statements in a logic which are evaluated against a trace or run representing the operations of an organization. The result of evaluation is either an affirmative answer to conformance, or a counterexample representing a subset of the operations of the organization and the specific law that is violated.

* This research was supported in part by NSF CCF-0429948, NSF-CNS-0610297, ARO W911NF-05-1-0158, and ONR MURI N00014-07-1-0907.

There are two important features of regulatory texts that need to be accomodated by a representation in logic. First, regulations convey constraints on an organization's operations, and these constraints can be obligatory (required) or permitted (optional). Second, statements in regulation refer to others for conditions or exceptions. An organization conforms to a body of regulation iff it satisfies all the obligations. However, permissions provide exceptions to obligations, indirectly affecting conformance. Our formulation of obligations and permissions follows the theory of Ross [5], and we will discuss the relationship to other theories (cf. [6]) in Section 3.1.

The central focus of this work is the function of regulatory sentences as conditions or exceptions to others. This function of sentences makes them dependent on others for their interpretation, and makes the translation to logic difficult. We call this the problem of *references to other laws*. In Section 2, we argue that a logic to represent regulation should provide mechanisms for statements to refer to others. We provide motivation using examples from the FDA CFR. We discuss how these sentences can be represented in a logic without references, and conclude that this would make the translation difficult.

We then turn to the task of defining a logic that lets statements refer to and reason about others. In Section 3.1, we define a trace or run-based representation for the operations of an organization, and a predicate-based linear temporal logic (PredLTL) to make assertions about runs. PredLTL is extended to express two kinds of normative statements (obligations and permissions), leading to a formal definition of conformance.

In Sections 3.2 and 3.3, we extend PredLTL to allow references between laws thereby making permissions relevant to conformance. Specifically, we introduce *an inference predicate*, whose interpretation is determined by inferences from laws. The justifications in default logic [7] can be cast as an instance of this predicate. Default logic has been used in computing extensions to a theory, in the manner of logic programs [8, 9]. In conformance checking, we need to separate two uses of statements: (a) extending a theory (the regulation), and (b) determining facts about an organization. This separation is achieved using the inference predicate. Statements are evaluated using the fixed points of an appropriate function, based on a technique used in Kripke's theory of truth [10]. An axiomatization is discussed in Section 3.4.

Section 4 concludes with a discussion of related and future work.

2 Motivation

In this section, we argue that a logic to represent regulation should provide a mechanism for sentences to refer to others. We discuss shortened versions of sentences from the CFR Section 610.40, which we will use as a running example throughout the paper. Consider the following sentences:

(1) Except as specified in (2), every donation of blood or blood component must be tested for evidence of infection due to Hepatitis B.

(2) You are not required to test donations of source plasma for evidence of infection due to Hepatitis B.

Statement (1) conveys an obligation to test donations of blood or blood component for Hepatitis B, and (2) conveys a permission not to test a donation of source plasma

(a blood component) for Hepatitis B. To assess an organization's conformance to (1) and (2), it suffices to check whether "All non-source plasma donations are tested for Hepatitis B". In other words, (1) and (2) imply the following obligation:

(3) Every non-source plasma donation must be tested for evidence of infection due to Hepatitis B.

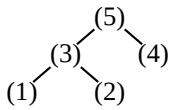
There are a variety of logics in which one can capture the interpretation of (3), as needed for conformance. Now suppose we have a sentence that refers to (1):

(4) To test for Hepatitis B, you must use a screening test kit.

The reference is more indirect here, but the interpretation is: "If (1) requires a test, then the test must be performed using a screening test kit". A bloodbank is not prevented from using a different kind of test for source plasma donations. (4) can be represented by first producing (3), and then inferring that (3) and (4) imply the following:

(5) Every non-source plasma donation must be tested for evidence of infection due to Hepatitis B using a screening test kit.

It is easy to represent the interpretation of (5) directly in a logic. However, (5) has a complex relationship to the sentences from which it was derived, i.e., (1), (2) and (4). The derivation takes the form of a tree:



To summarize, if one wishes to use a logic with no support for referring to other sentences, derived obligations must be created manually. We argue that the manual creation of derived obligations is impractical in terms of the amount of effort involved. We give two (pragmatic) reasons. First, the derived obligation can become very complex. The full version of statement (1) in the CFR contains six exceptions, and these exceptions in turn have statements that qualify them further. It is difficult to inspect a derived obligation, and determine if it captures the intended interpretation of the sentences from which it came. Second, references between laws are frequent, amplifying the effort in creating a logic representation. In [11], we discuss lexical statistics which suggest that references are a common way of establishing relationships between sentences in the CFR, and [12, 3] point out their frequency in other bodies of regulation.

We advocate an approach that allows us to introduce references into the syntax of the logic, and resolve references during evaluation.

3 Representing Regulatory Documents in Logic

In this section, we extend linear temporal logic (LTL) to distinguish between obligations and permissions, and allow references between statements. We begin, in Section 3.1, by representing a bloodbank as a run or trace. LTL is extended to distinguish between

obligations and permissions, leading to definitions of conformance. We then extend the logic to allow sentences to refer to others. Section 3.2 gives an informal example-driven account, and Section 3.3 provides a formal account. In Section 3.4, we discuss an axiomatization.

Sections 3.1 is intended as background, in which we discuss several underlying assumptions. Our goal is to focus on the problem of references, and to treat the representation of obligations and permissions as an important but orthogonal issue.

3.1 Predicate-based Linear Temporal Logic (PredLTL)

Representing regulated operations: Given the need to demonstrate conformance to the regulation in case of an audit, regulated organizations such as bloodbanks keep track of their operations in a database, for example, donor information and the tests they perform. Such a system can be thought of abstractly as a relational structure evolving over time. At each point in time (state), there are a set of objects (such as donations and donors) and relations between the objects (such as an association between a donor and her donations). The state changes by the creation, removal or modification of objects. We represent this as a run.

Definition 1 (A Run of a System). *Given a set O (of objects) and countable sets $\Phi_1, ..., \Phi_n$ (where Φ_j is a set of predicate names of arity j), a run of a system $R(O, \Phi_1, ..., \Phi_n)$, abbreviated as R, is a tuple $(r, \pi_1, ..., \pi_n)$ where:*

- *$r : N \rightarrow S$ is a sequence of states. N is the set of natural numbers, and S is a set of states.*
- *$\pi_j : \Phi_j \times S \rightarrow 2^{O^j}$ is a truth assignment to predicates of arity j. Given $p \in \Phi_j$, we will say that $p(o_1, ..., o_j)$ is true at state s iff $(o_1, ..., o_j) \in \pi_j(p, s)$.*

Given a run R and a time $i \in N$, the pair (R, i) is called a point (statements in linear temporal logic are evaluated at points). Given the predicate names $(\Phi_1, ..., \Phi_n)$, the corresponding space of runs is denoted by $\mathcal{R}(\Phi_1, ..., \Phi_n)$, abbreviated as $\mathcal{R}$.

Conceivably, we could construct a state-transition diagram representing all possible behaviors of the system and explore conformance from the model checking perspective (e.g., [13]). We chose to restrict our attention to traces for two reasons. First, checking of traces is easier to explain, and all interesting theoretical and algorithmic aspects that we explore in this paper manifest themselves in trace checking. Second, many parts of the operations of an organization, such as a bloodbank, do not involve computers. A complete model of operations has to include a model of human users, which is a research problem in its own right that is well beyond the scope of this paper. However, if a finite-state model of an organization can be created, the propositional version of the logic developed here can be adapted to work with available model-checkers.

Representing the regulation: The logic that we define in this section is a restricted fragment of first-order modal logic. The restriction is that we allow formulas with free variables, but no quantification over objects. Formulas will be interpreted using the universal generalization rule, i.e., over all assignments to free variables. The restrictions are similar in spirit to the logic programing approaches to regulation [8, 9]. PredLTL is

less expressive than the variants of first-order logic used by [2, 4]. However, when references are added, the logic becomes more expressive than first-order logic (Section 3.3).

Definition 2 (Syntax). *Given sets $\Phi_1, ..., \Phi_n$ (of predicate names) and a set of variables X, the language $L(\Phi_1, ..., \Phi_n, X)$, abbreviated as L, is the smallest set such that:*

- $p(y_1, ..., y_j) \in L$ *where* $p \in \Phi_j$ *and* $(y_1, ..., y_j) \in X^j$.
- *If* $\varphi \in L$, *then* $\neg\varphi \in L$ *and* $\Box\varphi \in L$. *If* $\varphi, \psi \in L$, *then* $\varphi \wedge \psi \in L$.

Disjunction $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$ and implication $\varphi \Rightarrow \psi = \neg\varphi \vee \psi$ are derived connectives. The temporal operator is understood in the usual way: $\Box\varphi$ (φ holds and will always hold (globally)). $\Diamond\varphi$ (φ will eventually hold) is defined as $\neg\Box\neg\varphi$.

We now extend the syntax to express normative statements in a body of regulation, by distinguishing between obligations and permissions.

Definition 3 (Syntax of Regulation). *Given a finite set of identifiers ID, a body of regulation Reg is a set of statements such that for each $id \in ID$, there exist $\varphi, \psi \in L$ such that either: id.**o**: $\varphi \rightsquigarrow \psi \in Reg$, or id.**p**: $\varphi \rightsquigarrow \psi \in Reg$*

id.**o**: $\varphi \rightsquigarrow \psi$ (id.**p**: $\varphi \rightsquigarrow \psi$) is read as: "it is obligated (permitted) that the precondition φ leads to the postcondition ψ". The distinction between preconditions and postconditions corresponds to the distinction between input and output in input-output logic [14].

Definition 4 (Semantics). *Given a run $R = (r, \pi_1, ..., \pi_n)$, $i \in N$, $\varphi \in L$, and an assignment $v : X \rightarrow O$, the relation $(R, i, v) \models \varphi$ is defined inductively as follows:*

- $(R, i, v) \models p(y_1, ..., y_j)$ *iff* $(o_1, ..., o_j) \in \pi_j(p, r(i))$ *where* $o_k = v(y_k)$ *if* $y_k \in O$.
- *The semantics of conjunction and negation is defined in the usual way.*
- $(R, i, v) \models \Box\varphi$ *iff for all* $k \geq i : (R, k, v) \models \varphi$

We extend the semantic relation to regulatory statements. We take $\models$ to stand for "conforms to":

- $(R, i, v) \models$ *id.***o**: $\varphi \rightsquigarrow \psi$ *iff* $(R, i, v) \models \varphi \Rightarrow \psi$ *($\Rightarrow$ is implication)*
- $(R, i, v) \models$ *id.***p**: $\varphi \rightsquigarrow \psi$. *Runs vacuously conform to permissions. Permissions will become relevant when references from obligations are present (Section 3.2).*

Consider again our example from Section 2. We use three predicates defined as follows. $d(x)$ is true iff x is a donation. $sp(x)$ is true iff x consists of source plama. $test(x)$ is true iff x is tested for Hepatitis B. Statement (3) is represented as:

3.**o**: $d(x) \wedge \neg sp(x) \rightsquigarrow \Diamond test(x)$

Statement (2) is be represented as: 2.**p**: $d(y) \wedge sp(y) \rightsquigarrow \neg\Diamond test(y)$. However, statement (1) cannot be represented directly.

We will now define conformance, and then discuss the various definitions in the context of related work. Given a run R, let $V(R)$ denote the set of variable assignments. Conformance is defined using the notion of validity. A formula φ is valid at the point (R, i), denoted $(R, i) \models \varphi$, iff for all $v \in V(R)$: $(R, i, v) \models \varphi$. A formula φ is valid on R iff it is valid at all points, that is, $R \models \varphi$ iff for all $i \in N : (R, i) \models \varphi$.

Definition 5 (Run Conformance). *Given a body of regulation Reg and a run R representing the operations of an organization, we say that R conforms to the regulation iff for all obligations $id.\mathbf{o}$: $\varphi \rightsquigarrow \psi \in Reg$, we have $R \models id.\mathbf{o}$: $\varphi \rightsquigarrow \psi$.*

Discussion: The deontic concepts of obligation and permission are treated as properties of sentences. Only obligations matter for conformance. If a non-source plasma donation is not tested, there is a problem. On the other hand, a bloodbank may choose to test a donation of source plasma or not. In assessing conformance, the function of a permission is to serve as an exception to an obligation, and in this indirect manner it becomes relevant. We will give a semantics to this function of permissions in Section 3.2. Such a treatment of permissions has its basis in the legal theory of Ross [5].

Ross' approach to permission is by no means the only one. Theories have distinguished between various kinds of permission (cf. [6]), the most common distinction being that of positive and negative permission. We discuss the analysis by Makinson and van der Torre [15]. φ is said to positively permitted iff it is explictly permitted by the laws, and φ is negatively permitted iff it is not forbidden. The key issue is whether positive permissions can give rise to violations. In regulations phrased exclusively in terms of permissions, it is desirable to say that *if φ denotes a "relevant" condition which is not explicitly permitted, then it should not hold in conforming implementations*. While this has been analysed as a property of permission, following Ross, we take such violations as arising from an implicit obligation, i.e., the italicized clause. This implicit obligation can be represented using the techniques we discuss in Section 3.2, provided that the relevance of the condition is known.

In the formulation here, obligations and permissions are top-level operators and cannot be negated. This restriction can be removed by treating obligation and permission as KD modalities (c.f. [16]), and using a many-valued interpretation to decide if a run belongs to the set of ideal runs. However, we avoid this to simplify presentation. A more crucial restriction is that iterated deontic constructs cannot be expressed directly, i.e., sentences of the form "required to allow x" or "allowed to require x.". One has to decide what top-level obligations or permissions are implied by these constructs. To our knowledge, handling iterated constructs is an open problem in deontic logic [17].

3.2 References to Other Laws – An Informal Description

In this section, we give an informal account of *reference logic* (RefL), which is used to handle references. We extend the syntax of PredLTL with *an inference predicate* $\mathrm{by}_{\mathrm{Id}}(\varphi)$, where Id is a set of identifiers. $\mathrm{by}_{\mathrm{Id}}(\varphi)$ is read as "by the laws in Id, φ holds". There are two restrictions: (a) φ is a statement in PredLTL (Definition 2) and (b) the predicate $\mathrm{by}_{\mathrm{Id}}(\varphi)$ can appear only in preconditions of laws. These restrictions are similar to those that apply to justifications in default logic [7].

Consider again our example statements (1) and (2), which are represented in RefL as follows:

- 1.**o**: $d(x) \wedge \neg \mathrm{by}_{\{2\}}(\varphi(x)) \rightsquigarrow \Diamond test(x)$, and
- 2.**p**: $d(y) \wedge sp(y) \rightsquigarrow \neg\Diamond test(y)$

In the obligation above, the subformula $\mathrm{by}_{\{2\}}(\varphi(x))$ is understood as "by the law (2) the formula $\varphi(x)$ holds". It remains to define the formula $\varphi(x)$. Intuitively, this should be the negation of the postcondition of (1). In other words, if $\neg\Diamond test(x)$ follows from (2), then the postcondition of (1) need not hold. This gives us:

1.**o**: $d(x) \wedge \neg\mathrm{by}_{\{2\}}(\neg\Diamond test(x)) \rightsquigarrow \Diamond test(x)$

We interpret the predicate $\mathrm{by}_{\{2\}}(\neg\Diamond test(x))$, by letting formulas have output. In other words, when the precondition of an obligation or permission is true at a point, the point is *annotated* with the postcondition.

Time	Objects	Predicates	Annotations
1	o_1	$d(o_1)$, $sp(o_1)$, $\neg test(o_1)$	2: $\neg\Diamond test(o_1)$
2	o_1	$d(o_1)$, $sp(o_1)$, $\neg test(o_1)$	2: $\neg\Diamond test(o_1)$
	o_2	$d(o_2)$, $\neg sp(o_2)$, $\neg test(o_2)$	1: $\Diamond test(o_2)$
3	o_1	$d(o_1)$, $sp(o_1)$, $test(o_1)$	2: $\neg\Diamond test(o_1)$
	o_2	$d(o_2)$, $\neg sp(o_2)$, $\neg test(o_2)$	1: $\Diamond test(o_2)$

Table 1. A run and its annotations

Table 1 shows a run of a bloodbank augmented with annotations. First, an object o_1 is entered into the system. o_1 is a donation of source plasma ($d(o_1)$ and $sp(o_1)$ are true). When a donation is added, its test predicate is initially false. Then, an object o_2 is added, which is a donation but not of source plasma. In the third step, the object o_1 is tested. At this point, unless the run is extended to test o_2 as well, it does not conform with the regulation. We now discuss how the annotations are arrived at and used to assess the regulation.

We begin by defining an annotation. Given a run R, an assignment $v \in V(R)$, and $\varphi \in L$, $v(\varphi)$ is the formula obtained by replacing all variables x by the unique name for the object $v(x)$. We assume that all variables are free. Note that $v(\varphi)$ is equivalent to a propositional LTL formula, as the variables have been replaced by constant symbols. An annotation, id: $v(\varphi)$, is a propositional LTL formula associated with an identifier.

Given a point (R, i) and an assignment $v \in V(R)$, first we consider the permission 2.**p**: $d(y) \wedge sp(y) \rightsquigarrow \neg\Diamond test(y)$. If $(R, i, v) \models d(y) \wedge sp(y)$, then (R, i) is *annotated* with 2: $v(\neg\Diamond test(y))$. Otherwise, there is no annotation.

Since the precondition of statement (2) is true for the assignment of y to o_1, we have the annotation 2: $\neg\Diamond test(o_1)$ at all points. However, since o_2 is not a donation of source plasma, there is no correponding annotation.

Now consider the formula $\mathrm{by}_{\{2\}}(\neg\Diamond test(x))$. This is evaluated as follows. We evaluate 2.**p**: $d(y) \wedge sp(y) \rightsquigarrow \neg\Diamond test(y)$ at (R, i) w.r.t. all variable assignments. Let ψ_2 be the conjunction of the annotations produced by the formula for (2).

$(R, i, v) \models \mathrm{by}_{\{2\}}(\neg\Diamond test(x))$ iff $\models \psi_2 \Rightarrow v(\neg\Diamond test(x))$

Notice that this requires a validity check in propositional LTL, which can be decided in space polynomial in the size of the formula [18].

Returning to the run in Table 1, the states are annotated with 2: $\neg\Diamond test(o_1)$ and $\models \neg\Diamond test(o_1) \Rightarrow \neg\Diamond test(o_1)$, since $\varphi \Rightarrow \varphi$ is a propositional tautology. So $(R, i, v) \models \text{by}_{\{2\}}(\neg\Diamond test(x))$ when $v(x) = o_1$.

We can evaluate 1.**o**: $d(x) \wedge \neg\text{by}_{\{2\}}(\neg\Diamond test(x)) \rightsquigarrow \Diamond test(x)$ similarly by annotating states with $\Diamond test(x)$ if the precondition holds. In Table 1, this results in an annotation of 1: $\Diamond test(o_2)$ on the appropriate states. If o_2 is never tested, the run will be declared non-conforming (by Definition 5), but the annotation will remain. This lets a law which depends on (1) draw the correct inference.

3.3 Reference Logic (RefL)

The semantic evaluation outlined in Section 3.2 works only when the references are acyclic, since an order of evaluation needs to be defined. To handle cycles, we adopt a fixed-point technique from Kripke's theory of truth [10]. The idea is to move to a three-valued logic where the third (middle) value stands for *ungrounded*. Initially, all statements are ungrounded and there are no annotations. Using an inflationary function, we add annotations until a fixed point in reached. In this section, we define this inflationary function and show that it has least and maximal fixed points. We begin by extending the syntax described in Section 3.1:

Definition 6 (Syntax of Preconditions). *Given sets $\Phi_1, ..., \Phi_n$ (of predicate names), a set of variables X, and a finite set of identifiers ID, the language $L'(\Phi_1, ..., \Phi_n, X, ID)$, abbreviated as L', is the smallest set such that:*

- *$p(y_1, ..., y_j) \in L'$ where $p \in \Phi_j$ and $(y_1, ..., y_j) \in X^j$.*
- *If $\varphi \in L'$, then $\neg\varphi \in L'$ and $\Box\varphi \in L'$. If $\varphi, \psi \in L'$, then $\varphi \wedge \psi \in L'$*
- *If $Id \subseteq ID$ and $\varphi \in L(\Phi_1, ..., \Phi_n, X)$ (Definition 2), then $\text{by}_{\text{Id}}(\varphi) \in L'$*

The syntax of regulatory statements (Definition 3) is modified so that the preconditions of laws are statements from L'. We use $id.\mathbf{x} : \varphi \rightsquigarrow \psi$ to stand for a normative statement (either obligation or permission). We now define an annotation:

Definition 7 (Annotation). *Given a run R, a set of identifiers ID, a body of regulation Reg and $v \in V(R)$, an annotation is a statement id: $v(\psi)$ such that $id \in ID$ and $id.\mathbf{x} : \varphi \rightsquigarrow \psi \in Reg$. The set of annotations is denoted by $A(R, ID, Reg)$, abbreviated A.*

Definition 8 (Annotation Function). *Given a run R, an annotation function $\alpha : N \rightarrow 2^A$ assigns a set of annotations to each point. We use $\alpha.Id(i)$ to denote the set of annotations id: $\psi \in \alpha(i)$ such that $id \in Id$.*

We will formalize the semantics using the fixed point technique outlined in [10]. Before we turn to the formal definitions, we sketch some of the key ideas involved.

Let us assume as given a run R. Statements in L' and Reg are divided into three classes corresponding to true ($\mathbf{T}(i, v)$), false ($\mathbf{F}(i, v)$) and ungrounded ($\mathbf{U}(i, v)$) for all times $i \in N$ and assignments $v \in V(R)$. Intuitively, $\mathbf{U}(i, v)$ is the set of statements that are waiting for the evaluation of another statement.

As we discussed in Section 3.2, to determine whether $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{T}(i,v)$, we need to check if there is a set of annotations which imply $v(\varphi)$. We construct the annotation function α such that for all assignments v, we have id: $v(\psi) \in \alpha(i)$ iff $\varphi \in \mathbf{T}(i,v)$ for some $id.\mathbf{x}: \ \varphi \rightsquigarrow \psi \in Reg$ and $id \in Id$. We will say that $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{T}(i,v)$ only if $\alpha.Id(i) \cup \{v(\neg\varphi)\}$ is not satisfiable.

To determine whether $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{F}(i,v)$, we need to ensure that there is no ungrounded statement that could make it true. To check this condition, we construct the annotation function α' such that id: $v(\psi) \in \alpha'(i)$ iff $\varphi \in \mathbf{T}(i,v) \cup \mathbf{U}(i,v)$ for some $id.\mathbf{x}: \ \varphi \rightsquigarrow \psi \in Reg$ and $id \in Id$. The condition for falsity w.r.t. α' is simply the negation of the condition for truth w.r.t. α. More formally, $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{F}(i,v)$ only if $\alpha'.Id(i) \cup \{v(\neg\varphi)\}$ is satisfiable.

When there are circular references, one cannot always evaluate a statement to be true or false. The Nixon-diamond problem (introduced in [7]) is a well-known example. We rephrase it in "legalese":

(6) Except as otherwise specified, Quakers must be pacifists.

(7) Except as otherwise specified, Republicans must not be pacifists.

These statements can be represented in RefL as follows:
6.**o**: $q(x) \wedge \neg\mathrm{by}_{\{6,7\}}(\neg p(x)) \rightsquigarrow p(x)$, and
7.**o**: $r(x) \wedge \neg\mathrm{by}_{\{6,7\}}(p(x)) \rightsquigarrow \neg p(x)$

Suppose we are given a state with an individual n (for Nixon), who is both quaker and republican, i.e., $q(n)$ and $r(n)$ hold. How should we evaluate the statements above? [10] suggests two answers to this question: (A) The statements are neither true or false (they are ungrounded). This corresponds to skeptical reasoning in non-monotonic logic. (B) Exactly one of $\mathrm{by}_{\{6,7\}}(p(n))$ and $\mathrm{by}_{\{6,7\}}(\neg p(n))$ is true, which leads us to conclude $p(n)$ (by (6)) or $\neg p(n)$ (by (7)) resply. This corresponds to credulous reasoning in non-monotonic logic.

In the semantics we give below, different answers correspond to different fixed points. We refer the reader to [10] for examples and discussion of the various possibilities with regard to fixed points. The choice of what to do when there are multiple fixed points depends on the application, and we discuss this issue further at the end of this section.

Definition 9 (Evaluation). *Given a run R and a body of regulation Reg, an evaluation is a tuple $E = (\mathbf{T}, \mathbf{F}, \mathbf{U})$, where $\mathbf{T}$, $\mathbf{F}$ and $\mathbf{U}$ are functions of the form $N \times V(R) \rightarrow 2^{L^+}$, where $L^+ = Reg \cup L'$. Furthermore, for all $i \in N$ and $v \in V(R)$, we have $\mathbf{T}(i,v) \cap \mathbf{F}(i,v) = \emptyset$ and $\mathbf{U}(i,v) = 2^{L^+} - (\mathbf{T}(i,v) \cup \mathbf{F}(i,v))$.*

Given an evaluation E, α_E is the annotation such that for all $i \in N$ and $id \in ID$, we have id: $v(\psi) \in \alpha_E(i)$ iff $\varphi \in \mathbf{T}(i,v)$, where $id.\mathbf{x}: \ \varphi \rightsquigarrow \psi \in Reg$. Similarly, α'_E is the annotation such that id: $v(\psi) \in \alpha'_E(i)$ iff $\varphi \in \mathbf{T}(i,v) \cup \mathbf{U}(i,v)$.

Definition 10 (Consistent Evaluation). *An evaluation E is consistent iff for all $i \in N$ and $v \in V(R)$, $\mathbf{T}(i,v) = \mathbf{F}(i,v) = \emptyset$, or $\mathbf{T}(i,v)$ and $\mathbf{F}(i,v)$ are sets such that:*

1. $p(x_1, ..., x_j) \in \mathbf{T}(i,v)$ *iff* $(v(x_1), ..., v(x_j)) \in \pi_j(p, r(i))$
   $p(x_1, ..., x_j) \in \mathbf{F}(i,v)$ *iff* $(v(x_1), ..., v(x_j)) \notin \pi_j(p, r(i))$

2. *If* $\phi \in \mathbf{T}(i,v)$ *and* $\psi \in \mathbf{T}(i,v)$*, then* $\phi \wedge \psi \in \mathbf{T}(i,v)$
   If $\phi \in \mathbf{F}(i,v)$ *or* $\psi \in \mathbf{F}(i,v)$*, then* $\phi \wedge \psi \in \mathbf{F}(i,v)$
   and similarly for negation and temporal operators
3. *If* $\varphi \Rightarrow \psi \in \mathbf{T}(i,v)$*, then id.***o***:* $\varphi \rightsquigarrow \psi \in \mathbf{T}(i,v)$
   If $\varphi \Rightarrow \psi \in \mathbf{F}(i,v)$*, then id.***o***:* $\varphi \rightsquigarrow \psi \in \mathbf{F}(i,v)$
   *id.***p***:* $\varphi \rightsquigarrow \psi \in \mathbf{T}(i,v)$*. Runs vacuously conform to permissions.*
4. *If* $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{T}(i,v)$*, then* $\alpha_E.Id(i) \cup \{v(\neg\varphi)\}$ *is not satisfiable.*
   If $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{F}(i,v)$*, then* $\alpha'_E.Id(i) \cup \{v(\neg\varphi)\}$ *is satisfiable.*

The set of all consistent evaluations for a run R *and regulation* Reg *is denoted by* $\mathcal{E}(R, Reg)$*, abbreviated* $\mathcal{E}$*.*

Observe that in consistent evaluations, if $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{T}(i,v)$, then $\alpha_E.Id(i) \cup \{v(\neg\varphi)\}$ is not satisfiable (Clause 4 in Definition 10). The converse need not be true.

Definition 11 (Partial Order). *Given evaluations* $E_1 = (\mathbf{T}_1, \mathbf{F}_1, \mathbf{U}_1)$ *and* $E_2 = (\mathbf{T}_2, \mathbf{F}_2, \mathbf{U}_2, \alpha_2)$*, we say that* $E_1 \leq E_2$ *iff for all* $i \in N$ *and* $v \in V(R)$*,* $\mathbf{T}_1(i,v) \subseteq \mathbf{T}_2(i,v)$ *and* $\mathbf{F}_1(i,v) \subseteq \mathbf{F}_2(i,v)$*.*

The pair $(\mathcal{E}, \leq)$*, where* $\mathcal{E}$ *is the set of consistent evaluations is a partially ordered set (poset).*

We now define the inflationary function whose fixed points we will be interested in.

Definition 12 (Inflationary function). *Given* $(\mathcal{E}, \leq)$*, the function* $\mathcal{I} : \mathcal{E} \to \mathcal{E}$ *is defined as follows. Given a consistent evaluation* $E_1 = (\mathbf{T}_1, \mathbf{F}_1, \mathbf{U}_1)$*,* $\mathcal{I}(E_1)$ *is the smallest consistent evaluation* $E_2 = (\mathbf{T}_2, \mathbf{F}_2, \mathbf{U}_2)$ *such that* $E_1 \leq E_2$*, for all* $i \in N$ *and* $v \in V(R)$*,* $\mathbf{T}_2(i,v) \neq \emptyset$*,* $\mathbf{F}_2(i,v) \neq \emptyset$*, and* E_2 *extends* E_1*.*

We say that E_2 *extends* E_1 *iff for all* $i \in N$ *and assignments* $v \in V(R)$*:*
If $\alpha_{E_1}(i) \cup \{v(\neg\varphi)\}$ *is not satisfiable, then* $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{T}_2(i,v)$
If $\alpha'_{E_1}(i) \cup \{v(\neg\varphi)\}$ *is satisfiable, then* $\mathrm{by}_{\mathrm{Id}}(\varphi) \in \mathbf{F}_2(i,v)$

It remains to show that $\mathcal{I}$ is well-defined, has maximal fixed points and a unique least fixed point. We give a brief sketch here, and refer the reader to [19] for detailed proofs.

Proposition 1. *Given* $(\mathcal{E}, \leq)$ *and* $E_1 \in \mathcal{E}$*, let* $\mathcal{E}_2 \subseteq \mathcal{E}$ *be the set of consistent evaluations such that* $E_2 \in \mathcal{E}_2$ *iff* $E_1 \leq E_2$*, for all* i *and* v*,* $\mathbf{T}_2(i,v) \neq \emptyset$*,* $\mathbf{F}_2(i,v) \neq \emptyset$*, and* E_2 *extends* E_1*. Then,* $\mathcal{E}_2$ *has a smallest element.*

The existence of fixed points is established using Zorn's lemma, which applies to chain-complete posets. Given the poset $(\mathcal{E}, \leq)$, a set $\mathcal{E}' \subseteq \mathcal{E}$ is called a chain (totally ordered set) iff for all $E_1, E_2 \in \mathcal{E}'$, we have $E_1 \leq E_2$ or $E_2 \leq E_1$. A poset is chain complete iff every chain has a supremum. The following can be shown:

Proposition 2. $(\mathcal{E}, \leq)$ *is a chain-complete poset.*

Lemma 1 (Zorn (c.f. [20])). *Every chain complete poset has a maximal element*

The existence of maximal fixed points is immediate from Zorn's lemma and the fact that $\mathcal{I}$ is inflationary, i.e., $E \leq \mathcal{I}(E)$. Let E^* be a maximal element in $\mathcal{E}$, since E^* is maximal and $E^* \leq \mathcal{I}(E^*)$ it follows that $E^* = \mathcal{I}(E^*)$.

To show the existence of a least fixed point, as [10] notes, we will need the observation that $\mathcal{I}$ *is monotonic*, i.e., if $E_1 \leq E_2$ then $\mathcal{I}(E_1) \leq \mathcal{I}(E_2)$. This can be shown by an argument similar to the proof of Proposition 1. With monotonicity, we obtain the following corollary to Zorn's lemma:

Corollary 1. *Given $E_1 \in \mathcal{E}$, let $\sigma(E_1)$ be the smallest set such that: (a) E_1 in $\mathcal{E}$, (b) if $E \in \sigma(E_1)$ then $\mathcal{I}(E) \in \sigma(E_1)$, and (c) if $C \subseteq \sigma(E_1)$ is a non-empty chain, then $E_{sc} \in \sigma(E_1)$, where E_{sc} is the supremum of C w.r.t. $\mathcal{E}$. Then:*

1. *$\sigma(E_1)$ is a chain whose supremum is a fixed point of $\mathcal{I}$*
2. *$\sigma(E_1)$ contains a unique fixed point*
3. *If $E_1 \leq E_2$, then $E_{s1} \leq E_{s2}$, where E_{s1} and E_{s2} are the suprema of $\sigma(E_1)$ and $\sigma(E_2)$ resply., and*
4. *$\mathcal{I}$ has a unique least fixed point.*

The first claim follows from a technique to prove Zorn's lemma [20]. The second and third claims follow from the first using monotonicity. And, for the last claim, consider the evaluation $E_0 = (\mathbf{T}_0, \mathbf{F}_0, \mathbf{U}_0)$, where for all $i \in N$, $v \in V(R)$, $\mathbf{T}_0(i, v) = \mathbf{F}_0(i, v) = \emptyset$, and $U_0(i, v) = 2^{L^+}$. Since $E_0 \leq E$ for all $E \in \mathcal{E}$, it follows from the third claim that $\sigma(E_0)$ is the least fixed point. The results are summarized in the following theorem, which provides a base for extending RefL with other inference predicates. We discuss the need for other predicates at the end of this section, and in Section 4.

Theorem 1. *Given the poset of consistent evaluations $(\mathcal{E}, \leq)$ and a function $\mathcal{I} : \mathcal{E} \rightarrow \mathcal{E}$ which is inflationary and monotonic, $\mathcal{I}$ has a least fixed point and a maximal fixed point.*

We mention the upper and lower bounds for the complexity of conformance checking w.r.t. the least fixed point. Given a run R and regulation Reg, we say that $R \models Reg$ iff all obligations are valid in R at the least fixed point. R is assumed to be finite in two ways: (a) The set of objects O is finite, and (b) There exists n, such that for all $j \geq n$, $r(n) = r(j)$, i.e., R eventually reaches a stable state.

Lemma 2 (Upper Bound). *Given a finite run R and regulation Reg, $R \models Reg$ can decided in EXPSPACE (space exponential in the size of Reg)*

The upper bound is obtained by turning Corollary 1 into a decision procedure. We start with the evaluation E_0, and apply $\mathcal{I}$ until a fixed point is reached. The worst-case size of the satisfiability tests are exponential in the size of the regulation. Since testing satisifiablity for propositional LTL is PSPACE-complete [18], applying $\mathcal{I}$ requires EXPSPACE. For the fragment of LTL discussed in this paper (using only $\Box$) satisfiability is NP-complete [18], and $R \models Reg$ can be decided in EXPTIME.

Lemma 3 (Lower Bound). *Given a finite run R and regulation Reg, $R \models Reg$ is hard for EXPTIME (time exponential in the size of Reg)*

The lower bound is shown by a reduction from first order logic enriched with a least fixed point predicate (the system YF in [21]). With circular references, we can encode reachability computations that cannot be expressed in first order logic.

Discussion: We now discuss some options in defining conformance, depending on the needs of the application. The sections of the FDA CFR that we have examined can be formalized so that there is a unique fixed point, and conformance is simply the satisfaction of obligations at this fixed point.

However, examples discussed in the literature suggest that it may not be desirable to always have a unique fixed point. A well-known example is that of contrary-to-duty (CTD) obligations (c.f. [16]). CTD obligations are those that arise when other obligations have been violated. Prakken and Sergot [16] point out an inflexibility in casting CTD structures as an instance of non-monotonic reasoning. We outline how this inflexibility can be avoided, using alternate definitions of conformance. Consider the following example from [14] (similar to one in [16]): *The cottage must not have a fence or a dog. If it has a dog, then it must have both a fence and a warning sign*. The question is what are the obligations when the cottage has a dog. We discuss two possible solutions.

The first solution is to treat the CTD norm as an exception to the first:

1.**o**: $\neg\mathrm{by}_{\{2\}}(f \vee d) \rightsquigarrow \neg(f \vee d)$ and 2.**o**: $d \rightsquigarrow f \wedge w$

The propositions f, d and w correpond to the cottage having a fence, dog and warning sign resply. Since there is a dog, the precondition of the second law is true, and this leads to the precondition of the first law being false. So if $f \wedge w$ holds, there is no violation. However, as [16] points out, it may be useful to detect that the situation is worse than the one in which there is no dog. In the second solution, we represent the laws as excluding each other, i.e., we conjoin $\neg\mathrm{by}_{\{1\}}(\neg(f \wedge w))$ to the precondition of the second law. At the least fixed point, both obligations are ungrounded, and we get two maximal fixed points – one in which $\neg(f \vee d)$ is obligated, and one in which $f \wedge w$ is obligated. Since d holds, there is a violation w.r.t. the former fixed point. In a scenario where there is no dog, a unique fixed point is obtained.

Our analysis of CTD structures achieves the same effect as the analyses in [16, 14]. However, [16, 14] characterize the CTD norm as presupposing the violation of the other, and then revising the situation. In future work, we plan to investigate predicates that capture this presuppositional analysis more directly.

3.4 Axiomatization

As we discussed in the context of Lemma 3, RefL contains first order logic enriched with a least fixed point predicate. It can be shown that the validity problem is Π^1_1-hard, and as a result, it cannot be recursively axiomatized. In this section, we briefly discuss an axiomatization of the propositional fragment of L' (the language of preconditions).

We assume as given a fixed finite domain of quantification, and replace variables by identifiers for domain elements. Given a set of identifiers ID, a propositionalized body of regulation has one or more statements of the form $id.\mathbf{x} : \varphi \rightsquigarrow \psi$ for each $id \in ID$. For example, the presence of $id.\mathbf{x} : \varphi_1 \rightsquigarrow \psi_1$ and $id.\mathbf{x} : \varphi_2 \rightsquigarrow \psi_2$ corresponds to different assignments to the variables.

To simplify presentation, we will assume that the references in the regulation are acyclic. This lets us obtain a unique fixed point and restrict attention to a two-valued logic. We discuss the general case at the end of this section.

A1 All substitution instances of propositional tautologies
A2 $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$
A3 $\Box\varphi \Rightarrow \varphi \wedge \Box\Box\varphi$
R1 From $\vdash \varphi \Rightarrow \psi$ and $\vdash \varphi$, infer $\vdash \psi$
R2 From $\vdash \varphi$ infer $\vdash \Box\varphi$

We characterize the inference predicate by the laws it refers to. To axiomatize $\mathrm{by}_{\mathrm{Id}}(\varphi)$, we need to reason about provability in the language L (propositional LTL). We say that $\varphi \in L$ is is provable (denoted $\vdash_L \varphi$) iff it is an instance of the axioms A1-A3, or follows from the axioms using the rules R1 and R2. Crucially, we will use the negation of provability in the premise of a rule. Similar mechanisms have been used to axiomatize default logic, e.g., in [22], satisfiability is used in the premise of a rule, and in [23], a modal language is augmented with an operator for satisfiability.

We begin by developing some notation. Given a set of regulatory statements $F = \{id_1.\mathbf{x} : \varphi_1 \rightsquigarrow \psi_1, ..., id_n.\mathbf{x} : \varphi_n \rightsquigarrow \psi_n\}$, let $F_{pre} = \{\varphi_1, ..., \varphi_n\}$ be the set of preconditions, $F_{post} = \{\psi_1, ..., \psi_n\}$ be the set of postconditions, and $F_{id} = \{id_1, ..., id_n\}$ be the set of identifiers. Given a finite set of formulas Γ, we denote the conjunction by $\bigwedge \Gamma$. The conjunction of the empty set is identified with $\top$ (a tautology). We use two rules for the inference predicate:

R3 For all $F \subseteq Reg$ with $F_{id} \subseteq Id$, from $\vdash_L \bigwedge F_{post} \Rightarrow \phi$, infer $\vdash \bigwedge F_{pre} \Rightarrow \mathrm{by}_{\mathrm{Id}}(\phi)$
R4 For all $\psi \in L'$, if for all $F \subseteq Reg$ with $F_{id} \subseteq Id$, either $\nvdash_L \bigwedge F_{post} \Rightarrow \phi$, or $\vdash \psi \Rightarrow \neg \bigwedge F_{pre}$, then infer $\vdash \psi \Rightarrow \neg\mathrm{by}_{\mathrm{Id}}(\phi)$.

Informally, R3 says that $\mathrm{by}_{\mathrm{Id}}(\phi)$ is true, if there exists a set of laws whose postconditions imply ϕ, and whose preconditions are true. R4 says that $\mathrm{by}_{\mathrm{Id}}(\phi)$ is false, if one of the preconditions is false for all sets of laws whose postconditions imply ϕ. In particular, if $\nvdash_L \bigwedge F_{post} \Rightarrow \phi$ for all appropriate subsets, then $\vdash \top \Rightarrow \neg\mathrm{by}_{\mathrm{Id}}(\phi)$, and using R1, $\vdash \neg\mathrm{by}_{\mathrm{id}}(\phi)$.

The rules have an equivalent axiomatic characterization, which is important in establishing completeness. Given $\phi \in L$, let $\mathcal{F}_{(Id,\phi)}$ be the set of subsets ($F \subseteq Reg$ with $F_{id} \subseteq Id$) such that $F \in \mathcal{F}$ iff $\vdash_L \bigwedge F_{post} \Rightarrow \phi$. Let $\Gamma_{(Id,\phi)}$ be the set such that $\neg \bigwedge F_{pre} \in \Gamma_{(Id,\phi)}$ iff $F \in \mathcal{F}_{(Id,\phi)}$. Finally, let $\Delta_{(Id,\phi)}$ be the set such that $\bigwedge F_{pre} \in \Delta_{(Id,\phi)}$ iff $F \in \mathcal{F}_{(Id,\phi)}$.

Proposition 3. *The following are provable:*

1. $\vdash \bigwedge \Gamma_{(Id,\phi)} \Rightarrow \neg\mathrm{by}_{\mathrm{Id}}(\phi)$
2. $\vdash \mathrm{by}_{\mathrm{Id}}(\phi) \Rightarrow \bigvee \Delta_{(Id,\phi)}$

The first claim is an immediate consequence of R4. And, the second claim follows from the first by propositional reasoning. It is easy to show that the axioms A1-A3, together with Proposition 3, and the rules R1 and R2 imply the rules R3 and R4. The inference predicate behaves like a modality:

Proposition 4. $\vdash \mathrm{by}_{\mathrm{Id}}(\varphi \Rightarrow \psi) \Rightarrow (\mathrm{by}_{\mathrm{Id}}(\varphi) \Rightarrow \mathrm{by}_{\mathrm{Id}}(\psi))$

Completeness is be established by a standard pre-model construction (see [19] for details). We now discuss the general case, i.e., when there are circular references and multiple fixed points. In the presence of multiple fixed points, we can define validity w.r.t. all fixed points, the least fixed point, or maximal fixed points. The axioms and rules discussed here can be adapted to characterize valdity w.r.t. all fixed points [19]. However, we have not obtained a direct characterization of validity w.r.t. the least or maximal fixed points. [22] provides an axiomatization of these three notions of validity for default logic, by translating the default rules into an autoepistemic logic. A question of interest is whether the the translation procedure in [22] can be adapted for RefL.

4 Conclusions and Future Work

We have motivated and described a logic (RefL) that accomodates references between laws. RefL separates two uses of statements – drawing inferences from regulation, and determining facts about an organization. We believe that this separation is crucial to the application of conformance checking.

The inference predicate blends two ideas from logic programming. First, the Kripke-Kleene-Fitting semantics [24], which uses three values for negation in logic programs. In RefL, we place the burden on a predicate, rather than on negation. The advantage is that connectives can behave as they do in a many valued logic. Second, contextual logic programs [25] use operations to restrict the context from which inferences are derived. Referring to specific laws (via identifiers) gives us a fine-grained control of context.

RefL provides a staring point in bringing the advantages of non-monotonic reasoning to systems such as [2, 4]. [2] represents business contracts as SQL queries, and [4] uses first-order logic augmented with real time operators. The inference predicate can be added to these systems, provided that the existential quantification is relativized to either the preconditions or the postconditions. However, restrictions are needed to ensure that the satisfiability tests remain decidable. [3] discusses the importance of anlayzing references, but do not provide a formalization.

In this work, we have considered references to laws that appear in preconditions. There is also the need for references in postconditions. An obvious case is for laws that cancel obligations and permissions given by another, e.g., *if a donation is not used for transfusion, exemption (3) no longer applies*. A more speculative case can be made for iterated deontic constructs [17], e.g., “required to allow x”. We suggest that the semantics will involve representing agents who introduce laws that reason about each other, e.g., *You are required to (introduce laws that) allow a patient to see his records*.

On the computational side, our goal is to be able to scale up to runs with a large number of objects, and incorporate RefL into a runtime checking framework for LTL. In a companion paper [26], we identify a fragment of RefL motivated by a case study of the FDA CFR. The fragment assumes that $\mathrm{by}_{\mathrm{Id}}(\varphi)$ can be evaluated by using at most one of the laws referred to. This assumption allows us to replace satisfiability tests with tests of lower complexity, and lets us scale up to runs with a large number of objects. In this paper, we have focussed on formally characterizing the semantics and complexity of RefL, and in [26], we focus on optimizations that are needed in practice.

References

1. U.S. Food and Drug Administration: Code of Federal Regulations. http://www.gpoaccess.gov/cfr/index.html
2. Abrahams, A.: Developing and Executing Electronic Commerce Applications with Occurrences. PhD thesis, Univeristy of Cambridge (2002)
3. Breaux, T.D., Vail, M.W., Anton, A.I.: Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In: Proceedings of the 14th IEEE International Requirements Engineering Conference. (2006)
4. Giblin, C., Liu, A., Muller, S., Pfitzmann, B., Zhou, X.: Regulations Expressed as Logical Models (REALM). In Moens, M.F., Spyns, P., eds.: Legal Knowledge and Information Systems. (2005)
5. Ross, A.: Directives and Norms. Routlege and Kegan Paul (1968)
6. Boella, G., van der Torre, L.: Permissions and obligations in hierarchical normative systems. In: Proceedings of the 9th international conference on AI and law. (2003)
7. Reiter, R.: A logic for default reasoning. In: Readings in nonmonotonic reasoning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1987) 68–93
8. McCarty, L.T.: A language for legal discourse - i. basic features. In: Proceedings of ICAIL. (1989)
9. Sergot, M., F.Sadri, Kowalski, R., F.Kriwaczek, P.Hammond, Cory, H.: The british nationality act as a logic program. Communications of the ACM **29**(5) (1986) 370–86
10. Kripke, S.: Outline of a theory of truth. Journal of Philosophy **72** (1975) 690–716
11. Dinesh, N., Joshi, A., Lee, I., Sokolsky, O.: Logic-based regulatory conformance checking. In: Proceedings of the 14th Monterey Workshop. (2007)
12. Bench-Capon, T., Robinson, G., Routen, T., Sergot, M.: Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation. In: Proceedings of the 1st International Conference on AI and Law. (1987)
13. Holzmann, G.: The Spin model checker. IEEE Trans. on Software Engineering **23**(5) (1997) 279–295
14. Makinson, D., van der Torre, L.: Input/output logics. Journal of Philosophical Logic **29** (2000) 383–408
15. Makinson, D., van der Torre, L.: Permissions from an input/output perspective. Journal of Philosophical Logic **32**(4) (2003)
16. Prakken, H., Sergot, M.: Contrary-to-duty obligations. Studia Logica **57**(1) (1996) 91–115
17. Marcus, R.B.: Iterated deontic modalities. Mind **75**(300) (1966)
18. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logic. ACM **32** (1985) 733–49
19. Dinesh, N., Joshi, A., Lee, I., Sokolsky, O.: A default temporal logic for regulatory conformance checking. Technical Report MS-CIS-08-07, University of Pennsylvania (2008)
20. Rudin, W.: Real and Complex Analysis. McGraw-Hill Book Company (1987)
21. Vardi, M.: The complexity of relational query languages. In: STOC. (1982)
22. Lakemeyer, G., Levesque, H.: Towards an axiom system for default logic. In: Proceedings of the AAAI Conference. (2006)
23. Halpern, J., Lakemeyer, G.: Multi-agent only knowing. Journal of Logic and Compuation **11**(1) (2001)
24. Fitting, M.: A Kripke/Kleene Semantics for logic programs. Journal of Logic Programming **2** (1985)
25. Monteiro, L., Porto, A.: A language for contextual logic programming. In Apt, K., de Bakker, J., Rutten, J., eds.: Logic Programming Languages: Constraints, Functions, and Objects. (1993)
26. Dinesh, N., Joshi, A., Lee, I., Sokolsky, O.: Checking traces for regulatory conformance. In: Proceedings of the Workshop on Runtime Verification. (2008)