

# Revising Institutions Governed by Institutions for Compliant Regulations

Thomas C. King<sup>1</sup>, Tingting Li<sup>2</sup>, Marina De Vos<sup>3</sup>,  
Catholijn M. Jonker<sup>1</sup>, Julian Padget<sup>3</sup>, and M. Birna van Riemsdijk<sup>1</sup>

<sup>1</sup>*Delft University of Technology*

{t.c.king-1, c.m.jonker, m.b.vanriemsdijk}@tudelft.nl

<sup>2</sup>*Imperial College London*

<sup>3</sup>*University of Bath*

tingting.li@imperial.ac.uk {mdv, jap}@cs.bath.ac.uk

**Abstract.** Institutions governing multi-agent systems (MASs) are a pervasive means to guide agents towards the aims of the MAS (e.g. collecting data) with regulations on the outcomes of agents' behaviour. Yet, wider organisations/governments often intend to guide the design of institutions governing MAS in meeting different aims (e.g. preserving the rights of agents). A pervasive means to guide the design of MAS-governing institutions (or any institution, for that matter) is to use institutions at higher tiers of governance (e.g. directives, constitutions) to regulate the regulations of institutions at lower tiers of governance (e.g. national legislation, software policies). A recent innovation has been an automated means to determine the compliance of a lower-tier institution's regulations with a higher-tier's. However, for a designer of a non-compliant institution there remains a dilemma: be punished for non-compliant regulations or arduously determine and rectify the underlying causes of non-compliance. In this paper we propose a way to automatically determine how to revise an institution to be compliant that also minimises the change in the regulations' outcomes thus keeping as closely as possible to the institution designers' original intentions.

**Keywords:** Multi-tier Institutions; Institution Revision; Institutional Compliance

## 1 Introduction

Legal institutions have long been used to govern Multi-Agent Systems (MAS) away from anarchic and uncoordinated behaviour towards a collaborative society through regulations that impose norms (obligations and prohibitions) on agents, leading to many frameworks for automated institutional reasoning (see [1] for a review). However, an institution governing an MAS is typically designed with only the global aim of the MAS (according to its stakeholders) in mind. Yet, institutions governing MASs can operate in the realm of governments and organisations with different aims to the MAS being governed, such as maintaining the rights of agents. Inevitably, tensions arise between the aims institutions guide MAS towards and the aims of the wider organisations and governments institutions reside in.

The social-world resolves such tensions by using institutions to govern other institutions, guiding institutional design towards wider aims with regulations on the outcomes

of other institutions' regulations. Known variously as multi-tier/multi-level/vertical governance [14], such governance structures consist in the tiering of institutions: a tier-1 institution governing an MAS by imposing obligations/prohibitions on agent behaviour, a tier-2 institution governing through regulating the outcomes of tier-1 regulation by obliging/prohibiting the imposition of specific obligations/prohibitions (i.e. imposing higher-order norms), and so on. In [10] we addressed the apparent lack of frameworks for institutions governing institutions with a formal and computational framework for the representation and reasoning of vertical governance structures which we call *multi-tier institutions*. The benefit being that (lower-tier) institutions can automatically be checked for compliance.

However, once non-compliance has been automatically determined, the problem remains for the designer of a non-compliant institution – determining how to revise the institution to be compliant and thus avoid any potential punishments for non-compliance (e.g. fines in the case of EU Directives). The difficulty is that there can be many causes of non-compliance due to the complexity of an institution and its multiple interacting rules. Thus, in this paper we use the framework in [10] for multi-tier institution representation and reasoning and propose an automated means to *revise* lower-tier institutions to comply with higher-tier institutions. To do so, we view revising an institution to be compliant as an Inductive Logic Programming problem where hypotheses (explanations for non-compliance) are sought. In order to solve the problem, we use abductive search implemented in Answer-Set Programming to abduce inductive explanations for non-compliance (ways to revise for compliance).

In the rest of this paper, we first introduce a running example of a two-tier institution in the domain of collecting audio data in Section 2. Then, we give some background in Section 3 on the formal multi-tier institution representation and reasoning, the computational multi-tier framework in Answer-Set Programming (ASP), and a brief re-introduction of Inductive Logic Programming (ILP) theory revision. In Section 4 we show how revising an institution to be compliant is an instance of an ILP problem, and show how we can resolve it by transforming a program representing an institution in ASP to a program for abducting revisions for compliance in ASP. The revision process is based on [12] for revising conflicting institutions adapted for revising non-compliant lower-tier institutions in multi-tier institutions with the following extensions: (i) creating or modifying existing rules for imposing higher-order norms, (ii) deleting existing rules and (iii) minimising the changes in the *consequences* of a revised institution compared to before revision. We further discuss differences with related work in Section 5 and conclude the paper in Section 6.

## 2 Running Example

Our running example is in the context of a system for crowdsourcing audio data from users using specialised cellphone apps, called a soundsensing system [15]. A tier-1 soundsensing institution is designed to guide the cellphone app users (i.e. an MAS) in collecting audio data. The soundsensing institution is described as follows:

### **Soundsensing Tier-1 Institution**

- Users are forbidden from turning their microphone off to ensure data is collected continuously.

- Users are obliged to provide their location on request to give the collected data location context.
- If a user violates a norm they are obliged to pay a fine.

In turn, the soundsensing institution is governed by a tier-2 governmental institution designed to meet different aims (e.g. maintaining agents’ rights), partly inspired by real-world regulations [18]:

#### **Governmental Tier-2 Institution**

- It is obliged that fines are only imposed on users after they violate a norm.
- When a user is in an area that forbids audio recording, it is forbidden to forbid them from turning their microphone off.
- It is forbidden to oblige children (users under the age of 14) to share their location (similar regulations can be found in the United States Government’s Child Privacy and Protection Act [18])

Putting these two institutions together, the tier-1 institution can be non-compliant for many reasons. Due to institution designer error, users might be obliged to pay a fine even when not violating a norm, and/or the tier-1 institution might not take into account areas where recording is forbidden or the possibility that users are children. Even if the tier-1 institution has, on the face of it, taken into account these factors, the interaction between different rules can mean all things considered it does not.

## **3 Background**

To provide context for this paper, we re-introduce the conceptualisation and operationalisation of individual and multi-tier institutions from [10] both formally and computationally. Then, we give an overview of ILP theory revision which we later use to formulate the problem of institution revision for compliance.

### **3.1 Formal Framework: Individual and Multi-tier Institutions**

As outlined previously, an individual legal institution acts as a mechanism to guide the behaviour of the system it governs. Institutions define a set of constitutive and regulative rules which respectively establish an institutional description and prescription of reality (see Searle’s counts-as relation [17]). Constitutive rules *describe* the system governed through creating institutional facts that can represent events caused by other events (e.g. entering a location which is private counts-as entering a private location), or they can represent changes to the institutional state (e.g. entering a private location causes an agent to be at a private location). Regulative rules *prescribe* what properties should hold/events should occur in a system by creating obligations and prohibitions in states (e.g. when requested an agent is obliged to share their location).

Conceptually, a multi-tier institution extends the notion of an individual institution governing an MAS to institutions governing institutions in a tiered structure. Each institutional tier governs the tier below, such that the first-tier imposes norms on what occurs and holds in an MAS (first-order norms), the second-tier norms on the norms imposed by the first (norms about first-order norms, i.e. second-order norms), and so on.

Formally, individual and multi-tier institutions are specified and reasoned about according to the following description. Firstly, the obligations/prohibitions which hold in states are represented as normative fluents describing an obligation/prohibition for an aim to occur before a deadline. Formally the grammar is  $n := obl(a, d) \mid pro(a, d)$  where  $a$  is the aim and  $d$  the deadline defined over a set of propositions  $Pr$  s.t.  $a, d \in Pr$ . The set of all expressible elements  $n$  is  $\mathcal{N}|_{Pr}$ . When  $Pr$  contains propositions denoting events and *non-normative* fluents, normative fluents about descriptive propositions are expressible allowing *first-order* norms to be expressed (e.g. Bertrand is obliged to share his location before leaving it:  $obl(share\_location(betrand), leave(betrand, street\_d))$ ). When  $Pr$  contains normative fluents, normative fluents about other normative fluents are expressible, allowing *higher-order* norms to be expressed (e.g. it is prohibited to oblige Bertrand to share his location until he turns 14:  $pro(obl(share\_location(betrand), leave(betrand, street\_d)), birthday(betrand, 14))$ ).

An individual institution specification, based on the InstAL framework [4] is a tuple  $\mathcal{I}|_{Pr} = \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$  (defined over a set of propositions  $Pr$  which in places we omit). The elements are: (i) The set of events  $\mathcal{E}$  that occur in the institution and bring about state change. These comprise observable events  $\mathcal{E}_{obs}$ , events that have an institutional meaning  $\mathcal{E}_{instruct}$  (e.g. an agent enters a *private* area) and events denoting a norm is discharged/violated  $\mathcal{E}_{norm}$ . (ii) The set of fluents  $\mathcal{F}$  that can hold in states. These comprise fluents used to describe the domain  $\mathcal{F}_{dom}$  and normative fluents  $\mathcal{F}_{norm} \subseteq \mathcal{N}|_{Pr}$ . (iii) An institutional event generation function  $\mathcal{G} : \mathcal{X} \times \mathcal{E} \rightarrow 2^{\mathcal{E}_{instruct}}$ . The function is conditional on the fluents that hold in a state (called a state condition represented with  $\mathcal{X} = 2^{\mathcal{F} \cup \neg \mathcal{F}}$  as a set of positive fluents that hold and negative fluents that do not hold in a state) and an event. (iv) An institutional state change function describing the fluents initiated and terminated from one state to the next conditional on the previous state and an event  $\mathcal{C} : \mathcal{X} \times \mathcal{E} \rightarrow 2^{\mathcal{F}} \times 2^{\mathcal{F}}$ . The codomain is a pair of sets  $\langle \mathcal{C}^+(\mathcal{X}, \mathcal{E}), \mathcal{C}^-(\mathcal{X}, \mathcal{E}) \rangle$  of initiated and terminated fluents. (v) The institution's initial state  $\Delta \subseteq \mathcal{F}$ .

Formally, a multi-tier institution is specified as a tuple  $\mathcal{M} = \langle \mathcal{T}, \mathcal{G}\mathcal{X}^i, \mathcal{C}\mathcal{X}^i \rangle$ . The components are: (i) A tiering of individual institutions  $\mathcal{T} = \langle \mathcal{I}^1|_{Pr^1}, \dots, \mathcal{I}^n|_{Pr^n} \rangle$  where we say an institution  $\mathcal{I}$  is in  $\mathcal{M}$  iff  $\exists i \in \mathbb{N} : \mathcal{I}^i = \mathcal{I}$ . (ii) A function  $\mathcal{G}\mathcal{X}^i$  for providing the normative events occurring during a state transition in one tier to the tier above for monitoring. (iii) A function  $\mathcal{C}\mathcal{X}^i$  for providing the normative fluents that hold in a state to the tier above for monitoring. The tiering of institutions restricts each institution in only imposing  $i$ th-order norms over the behaviour of the system it governs such that the normative fluents are defined over  $Pr^i$  which contains everything expressible in the tier below (i.e.  $Pr^2$  contains first-order norms and thus  $\mathcal{I}^2|_{Pr^2}$  imposes second-order normative fluents over  $Pr^2$ . For formal details see: [10]).

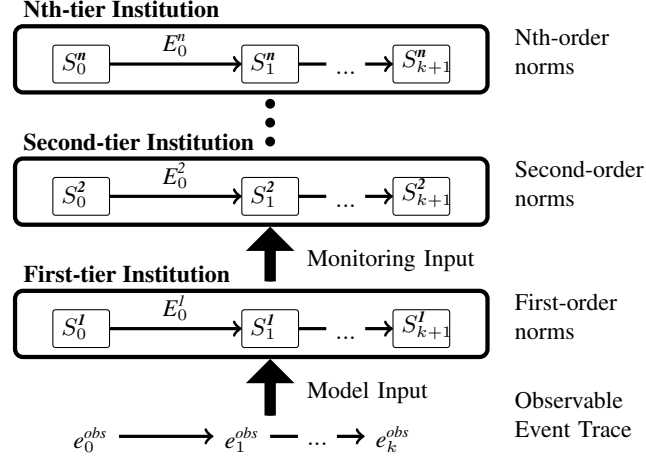
Table 1 formalises the running example as a multi-tier institution consisting of the soundsensing system's tier-1 institution and a governmental tier-2 institution (uppercase symbols stand for variables and for brevity we leave out the set of events and fluents for each institution). Both institutions consist of rules describing the domain (e.g. an agent entering a new location causes the agent to be at that location) and consider the location 'street b' to be private and the agent 'Bertrand' to be a child (see the initial states). The formalised example has three issues of non-compliance between the two institutions. Firstly, when an agent enters a new location this causes a generic norm

Soundsensing System Tier-1 Institution	Governmental Tier-2 Institution
$\mathcal{G}^1(\mathcal{X}, \mathcal{E}) :$ 1.1 $\langle \{at(Loc0, Ag0)\}, enter(Loc1, Ag0) \rangle \rightarrow \{leave(Loc0, Ag0)\}$ 1.2 $\langle \emptyset, viol(obl(share\_location(Ag0), leave(Ag0, Loc0))) \rangle \rightarrow \{norm\_violation(Ag0)\}$ 1.3 $\langle \emptyset, viol(pro(microphone\_off(Ag0), leave\_soundsensing(Ag0))) \rangle \rightarrow \{norm\_violation(Ag0)\}$ 1.4 $\langle \emptyset, enter(Ag0, Loc0) \rangle \rightarrow \{norm\_violation(Ag0)\}$ 1.5 $\langle \{private(Loc0)\}, enter(Ag0, Loc0) \rangle \rightarrow \{enter\_private(Ag0)\}$ 1.6 $\langle \{private(Loc0)\}, leave(Ag0, Loc0) \rangle \rightarrow \{leave\_private(Ag0)\}$ $\mathcal{C}^{1\uparrow}(\mathcal{X}, \mathcal{E}) :$ 1.7 $\langle \emptyset, enter(Loc0, Ag0) \rangle \rightarrow \{at(Ag0, Loc0)\}$ 1.8 $\langle \emptyset, \{request\_location(Ag0)\} \rangle \rightarrow \{obl(share\_location(Ag0), leave(Ag0, Loc0))\}$ 1.9 $\langle \emptyset, \{norm\_violation(Ag0)\} \rangle \rightarrow \{obl(pay\_fine(Ag0), leave\_soundsensing(Ag0))\}$ $\mathcal{C}^{1\downarrow}(\mathcal{X}, \mathcal{E}) :$ 1.10 $\langle \emptyset, \{leave(Loc0, Ag0)\} \rangle \rightarrow \{at(Loc0, Ag0)\}$ 1.11 $\langle \{child(Ag0)\}, \{birthday(Ag0, 14)\} \rangle \rightarrow \{child(Ag0)\}$ $\Delta^1 = \{private(street\_b), at(ada, street\_b), at(bertrand, street\_c), child(bertrand), pro(microphone\_off(ag0), leave\_soundsensing(ag0))\}$	$\mathcal{G}^2(\mathcal{X}, \mathcal{E}) :$ 2.1 $\langle \{at(Loc0, Ag0)\}, enter(Loc1, Ag0) \rangle \rightarrow \{leave(Loc0, Ag0)\}$ 2.2 $\langle \emptyset, viol(obl(share\_location(Ag0), leave(Ag0, Loc0))) \rangle \rightarrow \{norm\_violation(Ag0)\}$ 2.3 $\langle \emptyset, viol(pro(microphone\_off(Ag0), leave\_soundsensing(Ag0))) \rangle \rightarrow \{norm\_violation(Ag0)\}$ $\mathcal{C}^{2\uparrow}(\mathcal{X}, \mathcal{E}) :$ 2.4 $\langle \emptyset, enter(Loc0, Ag0) \rangle \rightarrow \{at(Ag0, Loc0)\}$ 2.5 $\langle \emptyset, disch(obl(pay\_fine(Ag0), leave\_soundsensing(Ag0))) \rangle \rightarrow \{obl(norm\_violation(Ag0), obl(pay\_fine(Ag0), leave\_soundsensing(Ag0)))\}$ 2.6 $\langle \{private(Loc0)\}, enter(Loc0) \rangle \rightarrow \{pro(pro(microphone\_off(Ag0), leave\_soundsensing(Ag0)), leave(Loc0))\}$ $\mathcal{C}^{2\downarrow}(\mathcal{X}, \mathcal{E}) :$ 2.7 $\langle \emptyset, \{leave(Loc0, Ag0)\} \rangle \rightarrow \{at(Loc0, Ag0)\}$ $\langle \{child(Ag0)\}, \{birthday(Ag0, 14)\} \rangle \rightarrow \{child(Ag0)\}$ $\Delta^2 = \{obl(norm\_violation(Ag0), obl(pay\_fine(Ag0), leave\_soundsensing(Ag0)), pro(obl(share\_location(bertrand), leave(Ag0, Loc0)), birthday(bertrand, 14))\} \cup \Delta^1$

**Table 1.** Formalisation of the tier-1 soundsensing institution and tier-2 governmental institution.

violation event in the soundsensing institution (1.3) due to designer error and which in turn initiates an obligation to pay a fine (1.9). However, the governmental institution only recognises actual norm violation events as causing a generic norm violation event (2.2 and 2.3) and obliges an actual norm is violated before an obligation to pay a fine is imposed (2.5 and  $\Delta^2$ ). Secondly, in the soundsensing institution agents are unconditionally prohibited from turning their microphone off ( $\Delta^1$ ), however the governmental institution prohibits such a prohibition when an agent enters a private location (2.6). Thirdly, when an agent is requested to provide their location the soundsensing institution obliges them to do so (1.8), but this is forbidden by the governmental institution if the agent is a child ( $\Delta^2$ ) such as Bertrand.

The operational semantics of a multi-tier institution, first presented in [10], allow such non-compliance to be determined by checking a multi-tier institution model for norm violations. Depicted in Figure 1, the model describes how each  $i$ th-tier institution evolves over time, as an event-state sequence, in response to the evolution of the tier



**Fig. 1.** Schematic view of a multi-tier institution model

below. The first-tier evolves in response to a trace of observable events that *could* occur in an MAS (i.e. produced for a pre-runtime check). Each tier above the first evolves in response to the event-state sequence of the institution they govern (i.e. the tier below). States contain domain fluents describing the MAS and normative fluents prescribing the events that should occur and fluents that should hold in the tier below (including other normative fluents). Each state transition is caused by events occurring in the institution from the previous state, which are in turn driven by the events and states from the tier below. If a normative fluent in a state is violated by an event or fluent in the tier below (including another normative fluent) a norm violation event occurs in the transition to the next state. Thus, model-checking can be used to compliance-monitor one institution with another by checking for higher-order norm violation events.

### 3.2 Computational Framework for Multi-Tier Institutions in ASP

The formal framework described is complemented by a corresponding computational framework in ASP (see [10]) for automatic compliance checking of lower-tier institutions with higher-tier institutions. ASP [2] is a non-monotonic logic programming language for representing problems where solutions to those problems, known as answer-sets, are computed according to the stable model semantics [8] using an answer-set solver (e.g. [7, 11]). An ASP program is built from first-order atoms which can be weakly negated with `not` or strongly negated with `—`. In an ASP program facts are of the form  $p_0$ . Rules are horn clauses of the form  $p_0 : -p_1, \dots, p_n$ , which states the head  $p_0$  is true when  $p_1, \dots, p_n$  are true. Constraints on answer-sets produced can be represented as  $:-p_1, \dots, p_n$ . meaning falsity is in the head of the rule and thus  $p_1, \dots, p_n$  is not true in any answer-set. Finally, choice constructs of the form  $l\{p_1, \dots, p_n\}u$  where  $l$  and  $u$  are positive integers state that at least  $l$  and at most  $u$  members of the set can arbitrarily be included in an answer-set (when omitted,  $l$  is 0 and  $u$  is infinity).

The computational framework consists of several components which we refer to later in this paper (i) an implementation of the operational semantics, the reasoning

$$\mathcal{M} = \langle \mathcal{T}, \mathcal{GX}^i, \mathcal{CX}^i \rangle, \mathcal{T} = \langle \mathcal{T}^1, \dots, \mathcal{T}^n \rangle, \forall i \in [n], (\mathcal{T}^i = \langle \mathcal{E}^i, \mathcal{F}^i, \mathcal{C}^i, \mathcal{G}^i, \Delta^i \rangle) :$$

$$\begin{aligned} \mathcal{T}^i &\Leftrightarrow \text{tier}(In, i). \text{inst}(In). \in P_{\mathcal{T}^i} \\ e \in \mathcal{E}_{obs}^i &\Leftrightarrow \text{evtype}(e, In, \text{ex}). \in P_{\mathcal{T}^i} \\ f \in \mathcal{F}^i &\Leftrightarrow \text{ifluent}(f, In). \in P_{\mathcal{T}^i} \\ e \in \mathcal{E}_{inact}^i &\Leftrightarrow \text{evtype}(e, In, \text{in}). \in P_{\mathcal{T}^i} \\ f \in \mathcal{F}^i &\Leftrightarrow \text{ifluent}(f, In). \in P_{\mathcal{T}^i} \\ \mathcal{C}^{i\uparrow}(X, e) = P &\Leftrightarrow \forall p \in P : \text{initiated}(p, In, I) : - \text{occurred}(e, In, I), EX(X, In, I). \in P_{\mathcal{T}^i} \\ \mathcal{C}^{i\downarrow}(X, e) = P &\Leftrightarrow \forall p \in P : \text{terminated}(p, In, I) : - \text{occurred}(e, In, I), EX(X, In, I). \in P_{\mathcal{T}^i} \\ \mathcal{G}^i(X, e) = E &\Leftrightarrow \forall e' \in E : \text{occurred}(e', In, I) : - \text{occurred}(e, In, I), EX(X, In, I). \in P_{\mathcal{T}^i} \\ f \in \Delta^i &\Leftrightarrow \text{holdsat}(f, In, I) : - \text{start}(I). \in P_{\mathcal{T}^i} \end{aligned}$$

**Table 2.** Multi-tier institution translation to ASP.

program  $\mathcal{P}_{reas}$  (ii) a program representing the trace of observable events used as input for producing multi-tier models, the timeline program  $\mathcal{P}_{time}$  and (iii) a representation in ASP of a multi-tier institution  $M = \langle \mathcal{T}, \mathcal{GX}^i, \mathcal{CX}^i \rangle$  according to the translation given in Table 2, which produces an ASP program  $\mathcal{P}_{\mathcal{T}^i}$  for each individual institution  $\mathcal{T}^i$  in the multi-tier institution  $\mathcal{M}$ . For brevity we leave out the details of the translation, but note that (i)  $In$  is a unique name for the institution  $\mathcal{T}^i$ , (ii)  $\text{initiated}(p, In, I)$  and  $\text{terminated}(p, In, I)$  means the fluent  $p$  is initiated/terminated at time  $I$  in institution  $In$ , (iii)  $\text{occurred}(e, In, I)$  means the event  $e$  occurs at time  $I$  in institution  $In$ , (iv)  $\text{holdsat}(f, In, I)$  means a fluent  $f$  holds at time  $I$  in institution  $In$ , (v)  $\text{start}(I)$  means  $I$  is the initial time interval according to the timeline program, and finally, (vi)  $EX(X, In, I)$  is shorthand for translating a state condition  $X \in \mathcal{X}^i$  into a corresponding set of ASP body literals  $\text{holdsat}(f, In, I)$  for all positive elements of  $X$  and not  $\text{holdsat}(f, In, I)$  for all negative elements of  $X$ .

### 3.3 Inductive Logic Programming: A brief overview

We view the problem of revising lower-tier institutions to be compliant with higher-tier institutions as a theory revision (TR) problem that can be solved using Inductive Logic Programming. ILP [16] is a machine learning technique concerned with the induction of logic theories that generalise (positive and negative) examples with respect to a prior background knowledge. In non-trivial problems it is crucial to define the search space accurately. This is done by a *language bias*, that can be expressed using the notion of mode declarations [16], describing the structure of the elements in the target theory. In the case presented here, we want to find ASP rules that contain certain elements in the head and body. So we will have head and body mode declarations.

An *ILP theory revision task* is a tuple  $\langle P, B, M \rangle$  where  $P$  is a set of conjunctions of literals, called *properties*,  $B$  is a normal program, called the *background theory*,  $M$  is a set of mode declarations describing the form that rules in the revised theory can take and  $s(M)$  is the set of rules adhering to  $M$ . A theory  $H$ , called a *hypothesis*, is an inductive solution for the task  $\langle P, B, M \rangle$ , if (i)  $H \subseteq s(M)$ , and (ii)  $P$  is true in all the answer sets of  $B \cup H$ .

Our approach to making a lower-tier institution in a multi-tier institution compliant is based on the introduction of new rules, and deleting and revising existing ones. As discussed in [5], non-monotonic inductive logic programming can be used to revise an existing theory. The key concept is that of *minimal revision*. In general, a TR system is biased towards the computation of theories that are similar to a given revisable theory. The difference between two programs  $T$  and  $T'$  is denoted as  $c(T, T')$ .

The theory  $T'$ , called a *revised theory*, is a *TR solution* for the task  $\langle P, B, T, M \rangle$  with distance  $c(T, T')$ , iff (i)  $T' \subseteq s(M)$ , (ii)  $P$  is true in all the answer sets of  $B \cup T'$ , (iii) if a theory  $S$  exists that satisfies conditions (i) and (ii) then  $c(T, S) \geq c(T, T')$ , (i.e. minimal revision).

## 4 Revising Institutions For Compliance

In this section we give the details of the paper’s main contribution: a system for revising a lower-tier institution to be compliant with a higher-tier institution in a multi-tier institution. In particular, we are interested in revising the institution which the system user (an institutional designer) has the power to effect change, which we call a *mutable* institution, to meet two properties:

- **Success** meaning that a formerly non-compliant institution for an event-trace is non-compliant is thereafter compliant for the same event trace. This means when normative fluents are obliged to be imposed they are, and conversely any prohibited normative fluents are not imposed.
- **Minimality** is a requirement for any revision to minimise the change in *consequences* of the new institution compared to the old one. That is, following changes to the institution the institution’s states are as close as possible to the states prior to the change(s) for a trace of events. To give an example, the soundsensing institution prohibits agents to turn their cellphone microphone off, whilst the governmental institution prohibits such a prohibition in areas deemed ‘private’. In this case, an institution revision can be successful by removing the soundsensing institution’s prohibition altogether, but only successful *and* minimal by removing the prohibition in just those cases where an agent is at a private location.

In order to revise a mutable institution to be compliant we instantiate it as an ILP theory revision task in Section 4.1. Then, we take a computational approach to solving the ILP theory revision task by performing *abductive* search in ASP [6]. Abductive search is achieved by transforming the mutable institution represented in ASP to an ASP representation encoding the space of ILP theory revisions and enabling different revisions to be tried. We describe our computational approach using ASP in Section 4.2, and the implementation and revision results for our running example in Section 4.3.

### 4.1 Revising Institutions to be Compliant is an ILP Theory Revision Task Instance

In this section, we define the revision for compliance task as an ILP revision task according to the revision for compliance requirements outlined previously. We begin by



formally defining the search space of possible revisions with *mode declarations*. Mode declarations define the literals that can appear in the head and body of rules. In the case of revising a mutable institution in a multi-tier institution, the mode declarations describe the valid rules for: generating non-normative institutional events, initiating and terminating domain fluents, and given the mutable institution is the  $i$ th-tier, initiating and terminating  $i$ th-order normative fluents (i.e. restricted to only initiating/terminating a normative fluent  $f$  if it is not in the language of norms  $\mathcal{N}|_{Pr^{i-1}}$  of the tier  $i-1$  below).

**Definition 1. Mode Declarations.** Let  $\mathcal{I}^i = \langle \mathcal{E}^i, \mathcal{F}^i, \mathcal{G}^i, \mathcal{C}^i, \Delta^i \rangle$  be a mutable institution for which  $In$  is a unique label. The mode declarations for  $\mathcal{I}^i$  are a pair  $M = \langle M^h, M^b \rangle$  where  $M^h$  is the set of head mode declarations and  $M^b$  the set of body mode declarations, defined as:

$$\begin{aligned} M^h &= \{ \text{initiated}(f, In, I), \text{terminated}(f, In, I) : f \in \mathcal{F} \setminus \mathcal{N}|_{Pr^{i-1}} \} \cup \\ &\quad \{ \text{occurred}(e, In, I) : e \in \mathcal{E}_{inst}^i \} \\ M^b &= \{ \text{holdsat}(f, In, I), \neg \text{holdsat}(f, In, I) : f \in \mathcal{F}^i \} \cup \\ &\quad \{ \text{occurred}(e, In, I) : e \in \mathcal{E}^i \} \end{aligned}$$

The set of *compatible rules* with the head and body mode declarations are also required to contain one event in the body and are defined as:

**Definition 2. Compatible Rules.** Let  $M = \langle M^h, M^b \rangle$  be the mode declarations for a mutable institution  $\mathcal{I}^i = \langle \mathcal{E}^i, \mathcal{F}^i, \mathcal{G}^i, \mathcal{C}^i, \Delta^i \rangle$ . An ASP rule  $l_0 : - l_1, \dots, l_n$  where  $n \in \mathbb{N}$  is compatible with  $M$  iff  $l_0 \in M^h$ ,  $\forall i \in [1, n] : l_i \in M^b$  and  $|\{l_1, \dots, l_n\} \cap \{ \text{occurred}(e, In, I) : e \in \mathcal{E}^i \}| = 1$ . The set of all compatible rules with  $M$  is  $s(M)$ .

Having described the search space of revisions, a theory revision task  $TR$  needs to be instantiated with the properties  $P$  that a solution must meet. These properties are typically positive examples (formulae that are true following a revision) and negative examples (formulae that are false following a revision). In our case we are only interested in supplying negative examples, stating that non-compliance is eradicated in a solution to  $TR$ . The negative examples in  $P$  are represented as ASP integrity constraints requiring a revised mutable institution is compliant with all higher-order norms it can violate – including those it does not violate before revision – ensuring revision does not cause further non-compliance.

**Definition 3. Compliance Properties.** Let  $\mathcal{I}^i$  be a mutable institution and  $\mathcal{I}^{i+1} = \langle \mathcal{E}^{i+1}, \mathcal{F}^{i+1}, \mathcal{C}^{i+1}, \mathcal{G}^{i+1}, \Delta^{i+1} \rangle$  be the institution with unique name  $In^{i+1}$  governing  $\mathcal{I}^i$  where  $i \in \mathbb{N}$ . The compliance properties for  $\mathcal{I}^i$  is the set of constraints:

$$P = \{ : - \text{occurred}(\text{viol}(n), In^{i+1}, I), \text{instant}(I) . : n \in \mathcal{F}_{norm}^{i+1} \}$$

We can now instantiate an ILP theory revision task, as a *compliance* theory revision task in a multi-tier institution according to the previous definitions:

**Definition 4. Compliance Theory Revision Task.** Let  $\mathcal{I}^i$  be a mutable institution in the multi-tier institution  $\mathcal{M}$ . An ILP theory revision task  $TR = \langle P, B, T, M \rangle$  is a compliance theory revision task for  $\mathcal{I}^i$  iff: (i)  $P$  is a set of compliance properties for  $\mathcal{I}^i$ , (ii)  $B$

is the normal program comprising (a) a multi-tier reasoning program  $\mathcal{P}_{reas}$ , (b) the timeline program  $\mathcal{P}_{time}$  and (c) the institution representation program  $\mathcal{P}_{\mathcal{I}^j}$  for each institution  $\mathcal{I}^j$  in  $\mathcal{M}$  apart from the mutable institution  $\mathcal{I}^i$ , (iii)  $T$  is the institution representation program  $\mathcal{P}_{\mathcal{I}^i}$  for the mutable institution  $\mathcal{I}^i$ , and (iv)  $M$  is the set of mode declarations for  $\mathcal{I}^i$ .

As outlined previously, we require solutions to theory revision to minimise the revision cost in order to remain as close to an institution designer's original intentions as possible. More precisely, the requirement is that the changed, mutable, institution's model for a composite trace contains as many similarities between states compared to before the changes were made (i.e. minimising the changes to consequences). We derive the cost of revision from the changes in consequences rather than the number of rule changes – as used in [12] – since due to non-monotonicity, as the changes in consequences between two versions of a mutable institution increases, the number of rule changes does *not* necessarily monotonically increase. The changes in consequences are the number of added and deleted fluents in the answer set for  $B \cup T$  compared to the answer-set  $B \cup T'$  for some revised institution  $T'$  (i.e. the symmetric set difference between the answer-sets for  $B \cup T$  and for  $B \cup T'$ ).

**Definition 5. Theory Revision Cost** Let  $TR = \langle P, B, T, M \rangle$  be a compliance theory revision task for a mutable institution  $\mathcal{I}$  with unique label  $In$ ,  $T'$  be a solution to  $TR$ ,  $ans$  be the answer-set for  $B \cup T$  and  $ans'$  be the answer-set for  $B \cup T'$  and  $\oplus$  be the set symmetric difference operation. The cost  $c(T, T')$  is defined as:

$$c(T, T') = |\{ f = \text{holdsat}(p, In, i) : i \in \mathbb{N}, f \in ans \oplus ans' \}|$$

## 4.2 Solving ILP Institution Revision in ASP

Based on [12] we use abductive search in ASP to solve an ILP theory revision task  $TR = \langle P, B, T, M \rangle$  instantiated as institutional revision for compliance. The approach we take is to transform the theory to be revised  $T$  (a mutable institution) into an ASP program where different changes to the theory can be tried/abduced (body literal and rule addition and deletion) that fit into the space of possibilities  $s(M)$ . We call this program the *revision* program  $\mathcal{P}_{rev}$ . The background theory  $B$  remains unchanged and provides both the unchangeable parts of the multi-tier institution, and multi-tier reasoning, allowing the effects of different revisions to be determined. The properties to be met,  $P$ , constrain any revisions found by the ASP program  $\mathcal{P}_{rev}$  to result in a compliant institution. The cost measure between a revisable  $T$  and revised theory  $T'$ ,  $c(T, T')$  is encoded as an ASP optimisation statement. Computing the answer-sets for these components as a single ASP program explores the search space, with each answer-set representing an outcome (revised theory) that meets the properties  $P$  and with those that minimise the difference (changes in consequences) ranked highest and presented to the user for selection. The advantage of this approach is that the representation and reasoning for the non-revisable portions of the multi-tier institution are encoded as the same ASP programs for the computational and revision framework requiring no re-implementation.

In order to go from a revisable theory  $T$  representing a mutable institution to a revision program  $\mathcal{P}_{rev}$ , we need to alter  $T$  in some way such that adding new rules and

New rules	Explanation
$l_0: -l_1, \dots, l_n, \text{rev}(In, i, \text{details}(\text{rDel})).$ $\{\text{rev}(In, i, \text{details}(\text{rDel}))\}.$	<i>Rule deletion:</i> Existing rules are extended with an abducible $\text{rev}(In, i, \text{details}(\text{rDel}))$ , which when included in an answer-set has the effect of deleting the rule with index $i$ .
$l_0: -l_1, \dots, l_{j-1}, \text{try}(i, j, B_-^+(l_j), l_j),$ $l_{j+1}, \dots, l_n.$ $\text{try}(i, j, B_-^+(l_j), l_j): -l_j,$ $\text{not rev}(In, i, \text{details}(\text{bDel}, j)).$ $\text{try}(i, j, B_-^+(l_j), l_j): -$ $\text{rev}(In, i, \text{details}(\text{bDel}, j)).$ $\{\text{rev}(In, i, \text{details}(\text{bDel}, j))\}.$	<i>Body literal deletion:</i> Each body literal $l_j$ of an existing rule is replaced with the literal $\text{try}/4$ for trying to delete the body literal $l_j$ . When the abducible $\text{rev}(In, i, \text{details}(\text{bDel}, j))$ is included in an answer-set the effect is to make the try literal true and thus effectively delete the literal $l_j$ , otherwise the try literal is only true when $l_j$ is true (effectively keeping $l_j$ ).
$l_0: -\text{rev}(In, i, \text{details}(\text{rAdd})), l_1, \dots, l_n.$ $\{\text{rev}(In, i, \text{details}(\text{rAdd}))\}.$	<i>Rule addition:</i> Including the abducible $\text{rev}(In, i, \text{details}(\text{rAdd}))$ has the effect of including the rule with index $i$ in the program.
$l_0: -l_1, l_2, \dots, l_n,$ $\text{extension}(i, l_0, l_{n+1}, B_-^+(l_{n+1})).$ $\text{extension}(i, l_0, l_{n+1}, B_-^+(l_{n+1})): -$ $\text{not rev}(In, i, \text{details}(\text{bAdd},$ $B_-^+(l_{n+1}), l_{n+1})).$ $\text{extension}(i, l_0, l_{n+1}, B_-^+(l_{n+1})): -$ $\text{rev}(In, i, \text{details}(\text{bAdd},$ $B_-^+(l_{n+1}), l_1), l_1.$ $\{\text{rev}(In, i, \text{details}(\text{bAdd},$ $B_-^+(l_{n+1}), l_{n+1}))\}.$	<i>Body literal addition:</i> Existing rules are appended with $\text{extension}/4$ predicates for each body mode literal a rule can be extended with. Including the abducible $\text{rev}(In, i, \text{details}(\text{bAdd}, \text{pos}, l_1))$ in an answer-set has the effect of extending the rule with index $i$ with the body literal $l_1$ (constraining the rule). That is, adding the revision predicate to an answer set makes the extension predicate true only when the literal with the specified variable bindings are true, effectively adding a constraint/body-literal to the rule. Otherwise, the extension predicate is always true (no constraint is tried for addition).

**Table 3.** Explanation of how abducible revision predicates can (re-)define institutional rules for finding revisions of the institution  $In$

changing existing rules can be tried by the new program with each answer-set corresponding to different revised theories. The approach we take, as in [6, 12], is to introduce *abducible* predicates which represent the different revision operations and are selected by the program for inclusion in answer-sets. If an abducible is selected for answer-set inclusion then the effect is to perform the revision operation the abducible represents. The abducibles have the form  $\text{rev}(In, i, \text{details}(\dots))$  conveying to the user the revision operation described in  $\text{details}(\dots)$  (e.g. a rule deletion operation) is carried out on a rule with label  $i$  in institution  $In$ . To give a simple example the rule  $l_0: -l_1$ . cannot be selected for deletion by an ASP program, but we can modify it to become  $l_0: -l_1, \text{not rev}(In, i, \text{details}(\text{rDel})).$  meaning if the abducible  $\text{rev}(In, i, \text{details}(\text{rDel}))$  is included by the program in an answer-set the effect is to delete the rule  $i$  by ensuring the body is never true. The selection of revision tuples for inclusion in an answer-set is encoded in the ASP revision program using the ASP *choice* construct of the form  $\{\text{rev}(In, i, \text{details}(\dots))\}.$

Each type of revision operation (rule and body literal addition and deletion) requires a different abducible and set of rules in the ASP revision program  $\mathcal{P}_{rev}$ . In Table 3 we describe the details of the different rules for trying revisions and the transformation from a revisable theory  $T$  to a revision program  $\mathcal{P}_{rev}$  using  $In$  to represent an institution's name,  $i$  to represent a rule identifier (e.g. an integer) and  $B_{-}^{+}(l)$  to represent whether a literal  $l$  is positive or negative.

Finally, the cost  $c(T, T')$  between two theories is encoded as an ASP optimisation constraint causing the ASP program to only present answer-sets that are minimal in the changes to consequences between  $T \cup B$  and  $T' \cup B$ , which we also extend with a secondary preference for revisions that *generalise* the institution (deleting body literals and rules) rather than specialising (adding new body literals and rules). The optimisation statement is given below where  $X@n$  represents the priority  $n$  of minimising the numerical value  $X$ ,  $\text{difference}/1$  measures the difference between the states in the answer-set for the institution before and after revision (in terms of added and removed fluents for each state),  $\text{rAdd}/1$  counts the rule additions,  $\text{bAdd}/1$  the body additions,  $\text{bDel}/1$  the body deletions and  $\text{rDel}/1$  the rule deletions.

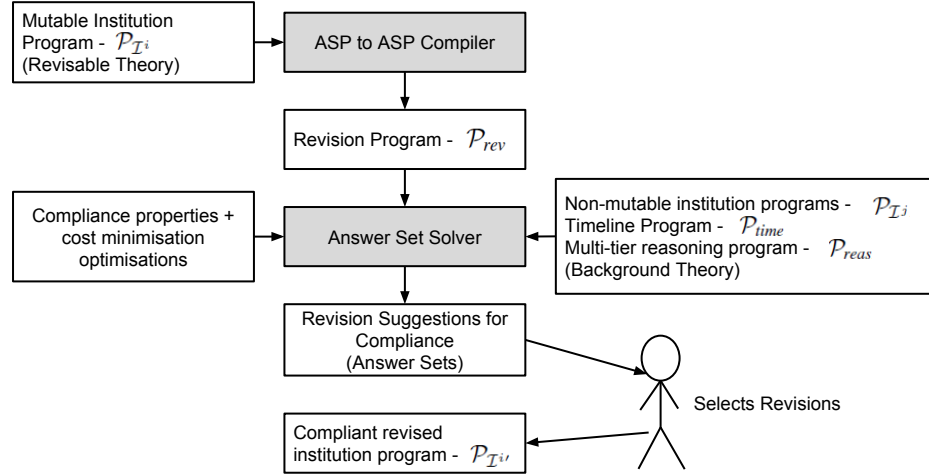
```
#minimize {D@5: difference(D); RA@4: rAdd(RA); BA@3: bAdd(BA); BD@2: bDel(BD);
RD@1: rDel(RD)}.
```

### 4.3 Implementation and Results

A prototype system for revising a lower-tier institution to be compliant with a higher-tier is implemented according to the description in the preceding sections<sup>1</sup>. The implementation is a compiler written in Java which, as depicted in Figure 2, takes as input the mutable institution the institution designer has the power to effect change represented in ASP (the mutable institution program  $\mathcal{P}_{Ti}$ ) and outputs a revision program  $\mathcal{P}_{rev}$ . The revision program is then put together with compliance properties to be met by revisions, revision cost minimisation optimisations and the background theory to remain unchanged (the non-mutable institutions, the timeline program and the multi-tier institution reasoning). An answer-set solver applied to the composition of these programs then produces minimal revision suggestions for compliance (answer sets). The suggestions are passed to a user who selects and applies a set of revisions, resulting in a compliant institution represented as an ASP program.

In addition to the system presented in this paper, the ASP compiler also addresses an apparent lack of re-usability of institutions (e.g. using the same institution for different sets of agents) due to their propositional nature. Rather than taking just propositional institutions as input, the compiler also takes *first-order* institution theories containing variables in the head and body of rules, together with bindings and monadic predicates to denote types. To give an example,  $\text{agent}(\text{ada})$  denotes ada is of type agent and  $\text{agent}(X)$  denotes the variable  $X$  is any ground term of type agent. Thus, a designer does not need to write a new propositional institution for the case where a new agent, Charles, joins the institutionalised society with all the norms and domain fluents that are

<sup>1</sup> The prototype, multi-tier reasoning in ASP and the examples used in this paper can be found at <https://sourceforge.net/projects/multitierinstitutionlearning/files/>



**Fig. 2.** Overview of using the implemented compiler and the multi-tier institution framework to resolve non-compliance.

about Charles. Instead, a fact `agent(charles)` can be added stating Charles is of type `agent`. In turn, the compiler takes these more re-usable first-order institution theories as input and outputs a first-order institution revision program that tries different variable bindings between head and body literals' variables of the same type.

For our running example, we have used our prototype compiler to produce a revision program for sub-sets of the compliance problem. That is, dividing the program up into smaller parts and resolving one case of non-compliance at a time for tractability, and testing all revision suggestions together at the end to confirm they are consistent. Some of the minimal and successful revisions found are given below (we keep to those we find most intuitive).

The first change suggested addresses the issue of non-compliance due to an obligation to pay a fine being imposed by the tier-1 institution when an agent enters a new area. Non-compliance occurs, because an agent entering a new area triggers a norm violation event in the first tier institution regardless of whether a norm has been violated, whilst the second tier obliges that a norm is genuinely violated before a fine is imposed. The revision suggestion is to delete the rule in the first-tier institution causing a norm violation event to occur when an agent enters an area:

```

occurred(norm_violation(Agent0), soundsensing, I) :- agent(Agent0), instant(I),
occurred(enter(Agent0, Location0), soundsensing, I), location(Location0).

```

The second issue is that children (people under the age of 14) are obliged to share their location when requested, but this is prohibited by the tier-2 institution. The following suggestion is one of several minimal changes found to ensure the non-compliant obligation is not imposed on children:

```

initiated(obl(share_location(Agent0), leave(Agent0, Location0)), soundsensing, I)
:- occurred(request_location(Agent1), soundsensing, I),
holdsat(at(Agent1, Location0), soundsensing, I),

```

```

not holdsat(child(Agent2), soundsensing, I), Agent0 = Agent1,
Agent0 = Agent2, agent(Agent0), agent(Agent1), agent(Agent2),
location(Location0), instant(I).

```

Finally, the tier-2 institution prohibits a prohibition on an agent to turn their microphone off when they are in a private area. Yet, the tier-1 institution always prohibits turning a microphone off until the agent leaves the system (the prohibition exists in the initial state). The revisions found are not to delete the rule initiating a prohibition in the tier-1 institution’s initial state, but instead, to terminate the prohibition when an agent enters a private area and then initiate it again when they leave. Although the revision adds two rules, it is minimal in the outcome of the tier-1 institution since there is still a prohibition on turning the microphone off in all other cases where it is allowed by the tier-2 institution:

```

terminated(pro(microphone.off(Agent0), leave.soundsensing(Agent0)), soundsensing, I)
:- occurred(enter.private(Agent2), soundsensing, I), agent(Agent2),
agent(Agent0), Agent0=Agent2, instant(I).
initiated(pro(microphone.off(Agent0), leave.soundsensing(Agent0)), soundsensing, I)
:- occurred(leave.private(Agent2), soundsensing, I), agent(Agent2),
agent(Agent0), Agent0=Agent2, instant(I).

```

## 5 Related Work

There has been much work on norm change in normative systems, however, as far as we are aware we are the first to propose a way to revise institutions to be *compliant* with other institutions in a multi-tier institution.

The most closely related work is by Li et al. [12, 13] who also uses abductive search in ASP to resolve an ILP *theory revision* task. Unlike us, their focus is on resolving norm conflicts between multiple institutions governing a group of agents (e.g. when an agent is prohibited to perform an action by one institution and obliged by another). In comparison, we focus on revising *non-compliance* between lower-tier and higher-tier institutions in a *multi-tier* institution. Our proposal is based on Li et al. and extended to revising an *ith*-tier institution by adding new rules or modifying/deleting pre-existing rules to impose *ith-order* norms. We also extend the work to revising with minimal changes in the *consequences* of a revised institution (as opposed to changed rules), finally we look at the creation and deletion of existing rules which in our running example provides more minimal changes in the consequences compared to rule modification.

Vasconcelos et al. [19] have proposed a technique for revising conflicting norms based on first-order unification. Their proposal provides a fine-grained way to revise obligation/permission/prohibition predicates’ terms. For example, an obligation to be in an area that overlaps with a prohibited area is revised by changing the obliged/prohibited areas for an agent to be in. In contrast to our work, their focus is on modifying the obligation/permission/prohibition predicates and not with adding/removing/modifying *rules* to meet a particular property (compliance between institutions in our case).

Governatori and Rotolo [9] propose a way to use a defeasible logic to modify legal systems by introducing new norms which derogate, abrogate and annul norms using defeasible rules. Central to their proposal is the idea of a legal system being *versioned* and having two timelines: the versioning timeline and the timeline of the legal system’s evolution (i.e. which norms are imposed and when). We only consider the latter timeline,

the evolution of an institution (in our case during pre-runtime model checking) and focus on diagnosing *causes* of non-compliance between institutions rather than assuming it is known what the new information (rules) is.

Finally, on the more conceptual and theoretical side, Boella et al. [3] look at how to classify different systems of norm change by investigating a set of rational norm change postulates. Specifically, they look at normative system change to incorporate new *conditional* norms in input/output logics and they investigate the set of consistent postulates for different input/output logics. Again, this work also presupposes which conditional norms should be added to the normative system/institution, thus any system meeting these postulates is quite different from our proposal.

## 6 Conclusions

In this paper we proposed an implemented automated system for revising a lower-tier institution's regulations to be compliant with the regulations of a higher-tier institution it is governed by. The proposal addressed a problem created by pervasive legal artefacts in the social world, where on the one hand institutions are used to govern other institutions in a vertical governance structure we call multi-tier institutions, creating the potential for non-compliant regulations. On the other hand, revising institutions' regulations to be compliant is non-trivial due to their inherent complexity.

Our proposal takes our previous formal and computational framework [10] for determining the compliance of institutions in multi-tier institutions. Then, viewing the problem of revising an institution to be compliant as an instance of an ILP (Inductive Logic Programming) theory revision task, we use abductive search in ASP based on [12] to solve the ILP theory revision task for compliance. Abductive search in ASP is performed by translating, using an implemented compiler, from an ASP representation of an institution that needs to be revised to be complaint where revisions cannot be tried and searched for, to an ASP representation where all possible revisions can be tried and thus revisions for compliance determined. Then, our system goes about finding revisions that are successful in resolving non-compliance and minimal in the changes to the institution's consequences thus keeping the regulations as close as possible to the institution designer's original intentions.

The system for revising institutions, for tractability, considers a fragment of the search space of revisions: modifying and deleting existing rules and extending the institution with a limited number of rules. The successful and minimal revisions that do exist (if any) within the space explored are guaranteed to be found. However, there may be more minimal revisions that result in a well-formed institution (according to our representation of institutions) outside of this space, but this space is bigger and takes longer to explore. We consider this a problem that is important to address, firstly with formal analysis of the complexity of the full problem. Secondly, by studying the applicability of various heuristics to the full search problem (e.g. genetic algorithms) which cannot guarantee a minimal solution is found (i.e. in the case of genetic algorithms instead converging on local optima) but can help resolve tractability issues. As yet, it is unclear which heuristics are appropriate and how they can be incorporated into ILP revision as abducible search in ASP, presenting an interesting challenge for future work.

**Acknowledgements** Thomas C. King is supported by TU Delft’s SHINE (<http://shine.tudelft.nl>) project. Authors would like to thank the anonymous reviewers of COIN@IJCAI 2015 for their helpful comments.

## References

1. G. Andrighetto, G. Governatori, P. Noriega, and L. van der Torre. Normative Multi-Agent Systems. *Dagstuhl Follow-Ups*, 4, 2013.
2. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge, 2003.
3. G. Boella, G. Pigozzi, and L. van der Torre. Normative framework for normative system change. In *Proceedings of AAMAS 2009*, pages 169–176, 2009.
4. O. Cliffe, M. De Vos, and J. Padget. Answer set programming for representing and reasoning about virtual institutions. *Computational Logic in Multi-Agent Systems*, 4371:60–79, 2006.
5. D. Corapi, O. Ray, A. Russo, A. K. Bandara, and E. C. Lupu. Learning rules from user behaviour. In *AIAI*, pages 459–468, 2009.
6. D. Corapi, A. Russo, and E. Lupu. Inductive logic Programming as Abductive Search. *Communications*, 7:54–63, 2010.
7. M. Gebser, B. Kaufmann, and R. Kaminski. Potassco: The Potsdam answer set solving collection. *AI Communications*, 24(2):107 – 124, 2011.
8. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, pages 1070 – 1080, 1988.
9. G. Governatori and A. Rotolo. Changing Legal Systems: Legal Abrogations and Annulments in Defeasible Logic. *Logic Journal of IGPL*, 18(1):157–194, 2010.
10. T. C. King, T. Li, M. De Vos, V. Dignum, C. M. Jonker, J. Padget, and M. B. V. Riemsdijk. A Framework for Institutions Governing Institutions. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, 2015.
11. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic (TOCL)*, 7(3):499 – 562, 2002.
12. T. Li. *Normative Conflict Detection and Resolution in Cooperating Institutions*. PhD thesis, University of Bath, 2014.
13. T. Li, T. Balke, M. De Vos, J. Padget, and K. Satoh. Legal Conflict Detection in Interacting Legal Systems. *DoCoPe@ JURIX*, 2013.
14. H. Liesbet and M. Gary. Unraveling the central state, but how? Types of multi-level governance. *American political science review*, 97(2):233–243, 2003.
15. H. Lu, W. Pan, N. Lane, T. Choudhury, and A. Campbell. SoundSense: scalable sound sensing for people-centric applications on mobile phones. *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 165–178, 2009.
16. S. Muggleton. Inverse entailment and Progol. *New generation computing*, 13:245–286, 1995.
17. J. R. Searle. What is an institution? . *Journal of Institutional Economics*, 1:1–22, 2005.
18. United States Federal Law. Children’s Online Privacy Protection Act, 1998.
19. W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman. Resolving conflict and inconsistency in norm-regulated virtual organizations. In *Proceedings of AAMAS ’07*, volume 5, pages 632–639, New York, New York, USA, 2007. ACM Press.