Fundamenta Informaticae XXI (2001) 1001–??
DOI 10.3233/FI-2016-0000
IOS Press

Dischargeable Obligations in the $\mathcal{S}$CIFF framework

Marco Alberti
Dipartimento di Matematica e Informatica
University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy

Marco Gavanelli
Dipartimento di Ingegneria
University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy

Evelina Lamma
Dipartimento di Ingegneria
University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy

Fabrizio Riguzzi
Dipartimento di Matematica e Informatica
University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy

Ken Satoh
Principles of Informatics Research Division, National Institute of Informatics
Chiyoda-ku, 2-1-2, Hitotsubashi, Tokyo 101-8430, Japan

Riccardo Zese
Dipartimento di Ingegneria
University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy

Abstract. Abductive Logic Programming (ALP) has been proven very effective for formalizing societies of agents, commitments and norms, in particular by mapping the most common deontic operators (obligation, prohibition, permission) to abductive expectations.

In our previous works, we have shown that ALP is a suitable framework for representing norms. Normative reasoning and query answering were accommodated by the same abductive proof procedure, named $\mathcal{S}$CIFF.

In this work, we introduce a defeasible flavour in this framework, in order to possibly discharge obligations in some scenarios. Abductive expectations can also be qualified as dischargeable, in the new, extended syntax. Both declarative and operational semantics are improved accordingly, and proof of soundness is given under syntax allowedness conditions.

Moreover, the dischargement itself might be proved invalid, or incoherent with the rules, due to new knowledge provided later on. In such a case, a discharged expectation might be reinstated and hold again after some evidence is given. We extend the notion of dischargement to take into consideration also the reinstatement of expectations.

Address for correspondence: marco.gavanelli@unife.it

The expressiveness and power of the extended framework, named $\mathcal{S}$CIFF$^{\mathcal{D}}$, is shown by modeling and reasoning upon a fragment of the Japanese Civil Code. In particular, we consider a case study concerning manifestations of intention and their rescission (Section II of the Japanese Civil Code).

Keywords: Normative reasoning, Abductive Logic Programming, Constraint Logic Programming, Dischargement of expectations.

1. Introduction

Supporting automated reasoning on legal rules and norms is very attractive, since it could help deriving the legal consequences of some happened actions in a fair and unbiased way, and obtain the results quickly, possibly in real-time. For these reasons, a lot of research has been devoted to defining suitable languages, knowledge bases, ontologies, and reasoning mechanisms. Notably, the British Nationality Act was formalized in Logic Programming [1]. The encoding of a set of norms into a formal languages, together with automatic mechanisms to reason about them, provides a so-called *normative system*. They are used, in particular, in multi-agent systems [2].

Deontic logic [3] is a valid tool, often used to represent knowledge in legal and normative reasoning. It provides an intuitive formalization of normative concepts, providing notions such as obligatory, permitted and forbidden actions.

Norms are often represented as implications: given some state of affairs (as precondition), there are some legal consequences that follow. Computational logics can then provide tools and proof-procedures to automatically reason about a given state of affairs, provide the deontic consequences, and possibly find violating parties. Being formalized in computational logic, all these tasks not only can be automatically performed, but also come with a guarantee of soundness and completeness, leveraging the properties of the available proof procedures.

Applications of computational logic to formalize norms include logic programming for the British Nationality Act [1], argument-based extended logic programming with defeasible priorities [4], defeasible logic [5]. Satoh et al.'s PROLEG [6] is a Prolog implementation of the Presupposed Ultimate Fact Theory of the Japanese Civil Code. The contract cancellation under the Japanese law was also formalized in computational logics [7].

In the multi-agent settings, organizational models have been defined [8, 9] exploiting Deontic Logic to specify the society norms and rules. EU research projects were devoted to formalizing norms for multiagent systems, like ALFEBIITE [10], IMPACT [11, 12] and SOCS. The latter, in particular, proposed various Abductive Logic Programming (ALP) languages and proof procedures to specify and implement both individual agents [13] and their interaction [14]; both approaches have later been applied to modeling and reasoning about norms with deontic flavours [15, 16].

In [17], the authors give an extensive survey of normative reasoning for multi-agent systems (MAS). Their emphasis is therefore more on normative systems in modern deontic logic, which – in the authors' words – "is much broader than the traditional modal logic framework of deontic logic". The introduction of normative issues in agent organizations, considering norms as key for the development of MAS, has led to Normative Multi-Agent Systems. Formal languages, often logic-based, to represent each agent's knowledge or modeling and designing a whole multi-agent system pave the

way to apply logic-based reasoning on these representations, for reasoning either about each single agent or the system in its whole. In [17] the authors survey two main approaches for normative reasoning: norm change and proof methods (but not necessarily implemented proof systems). Our work fits better in the latter category, it moves from a reasoning proof system, implemented as an abductive proof procedure named SCIFF, which was originally applied to model and reason upon interaction protocols in MAS modeled in a logic-based formal language as forward rules. The normative flavor of this language has been discussed in particular in [15]. Since norms impact on the practical behavior of individual agents, in the SCIFF language norms induce expectations, i.e. anticipations of the future course of events, together with an implicit goal to know whether the anticipated future eventuates (this is named fulfillment in the SCIFF framework). Nonetheless the normative state of a system can change, and it can be the case that expectations have to be withdrawn. This is the subject of this paper.

ALP is a powerful tool for knowledge representation and reasoning [18], for example it has been applied to represent ontologies [19, 20]. ALP is based on a declarative (model-theoretic) semantics and equipped with an operational semantics in terms of a proof-procedure. The IFF Proof-procedure [21] was proposed by Fung and Kowalski to support abductive reasoning also in presence of non-ground abducible literals. $\mathcal{S}$CIFF [14] extends IFF to deal with constraints *à la* Constraint Logic Programming (CLP) [22], optimization meta-predicates [23] and with both existentially and universally quantified variables in rule heads. The resulting system was used for modeling and implementing several knowledge representation frameworks, such as normative systems [24], accountable protocols for multi-agent systems [25], web service choreographies [26], and Datalog$^{\pm}$ ontologies [27, 28]. In particular, $\mathcal{S}$CIFF is particularly well suited for reasoning about norms because it features a concept of *expected behavior*, which is represented as a set of abducible atoms called *expectations* and produced by an abductive program reasoning on the current state of affairs (e.g., the current set of happened events). The concept of expectation as defined in $\mathcal{S}$CIFF supports reasoning on deontic concepts such as obligation, permission, forbidden actions [15].

This article is an extended and revised version of a previous conference paper [29]. In that work, we presented $\mathcal{S}$CIFF$^{\mathcal{D}}$, an extension of the $\mathcal{S}$CIFF framework that introduces a defeasible flavour in the norm portion of the framework, as a mechanism for discharging obligations. Rather than removing an abductive expectation representing obligation from the set of abduced atoms with a sort of contraction [30], we keep it, but we mark it as discharged to indicate that the lack of a fulfilling act is not a violation of the norms. Both declarative and operational semantics were extended accordingly, and a proof of soundness was given under syntax allowedness conditions. Thanks to that extension, we were better able to cope with real-life norms, even in the legal domain.

In this article, we further extend the work presented in [29]. In that work, we advocated the need to be able to discharge expectations, for example in cases in which, due to the unlawful behaviour of one agent, a contract was no longer valid, and all the expectations raised by the contract had to be removed. In real life, however, the very same invalidation of the contract could become invalid, and a normative reasoning system has to deal with such situations. If the invalidation is declared null, all expectations related to such contract return into existence, claiming to be fulfilled in order to avoid violations of the contract. The work in [29] was able to deal only with dischargement of expectations, and unable to deal properly with nullification of an invalidation, with the consequential reinstatement of expectations. In this article, instead, we propose a further extension of $\mathcal{S}$CIFF able to deal both

with dischargement and with reinstatement of expectations; the proposed extension does not limit the number of dischargements and reinstatements of an expectation, nor poses any limit in the nesting of nullifications of contracts and their invalidations.

For the proposed extension, we provide syntax, declarative semantics, and operational semantics, together with proof of soundness of the resulting proof-procedure with respect to its declarative semantics.

The paper is organized as follows. In Section 2, we first recall the $\mathcal{S}$CIFF language, also mentioning its declarative semantics and its underlying proof procedure, and discuss a case study from Section II of the Japanese Civil Code. Then, in Section 3, we introduce the $\mathcal{S}$CIFF$^{\mathcal{D}}$ syntax, with a novel abducible for discharging expectations (that is how obligations are mapped in $\mathcal{S}$CIFF); we also discuss the formalization of a further article from the Japanese Civil Code. Section 4 extends the declarative and operational semantics accordingly, and presents the proof of soundness for the extended framework. In Section 5 we discuss related work and in Section 6 we conclude the paper.

2. $\mathcal{S}$CIFF Language and Semantics

As a running example, we consider, throughout the paper, article 96 ("Fraud or duress") of the Japanese civil code. The Japanese civil code has been the subject of a series of competitions (COLIEE) and an English translation is available [31].

In order to model and discuss it, we first provide a description of the $\mathcal{S}$CIFF language; for a deeper discussion, we refer the reader to [15].

2.1. The $\mathcal{S}$CIFF Language

In $\mathcal{S}$CIFF, the agent behaviour is described by means of events (actual behaviour) and expectations (expected behaviour):

- events are ground atoms of the form $\mathbf{H}(Content, Time)$
- expectations are abducible literals of the following possible forms:
  - $\mathbf{E}(Content, Time)$: positive expectations, with a deontic reading of obligation (it is expected that $Content$ will happen at time $Time$);
  - $\mathbf{EN}(Content, Time)$: negative expectations, read as prohibition (it is expected that $Content$ will not happen at time $Time$);
  - $\neg\mathbf{E}(Content, Time)$: negation of positive expectation, or explicit absence of obligation (it is not expected that $Content$ will happen at time $Time$);
  - $\neg\mathbf{EN}(Content, Time)$: negation of negative expectation, or explicit permission (it is not expected that $Content$ will not happen at time $Time$).

  As will be clear from the semantics (Section 4), the latter two are used as a form of explicit negation of the former two.

$Content$ is a term that describes the event and $Time$ is a variable or a numeric constant representing the time of the event. In this work, we assume that $Content$ is not a variable (although it can be a term containing variables). CLP constraints can be imposed over variables; for time variables, they represent time constraints, such as deadlines. Variables occurring in abduced **E** literals are always existentially quantified, while those in **EN** literals can be universally quantified, if they do not occur also in other abduced atoms.

A $\mathcal{S}$CIFF program is a pair $\langle KB, \mathcal{IC} \rangle$. KB is a set of logic programming clauses $h \leftarrow body$ that can have literals (atoms and default negated atoms), expectations and other abducibles, but not events, in their bodies. KB is used to express domain specific knowledge. $\mathcal{IC}$ is a set of implications called Integrity Constraints, which implicitly define the expected behaviour of the interacting parties. Each Integrity Constraint (IC) in $\mathcal{IC}$ has the form $Body \rightarrow Head$, where $Body$ is a conjunction of events, expectations, abducibles and literals defined in KB, while $Head$ is a disjunction of conjunctions of expectations, abducibles, and CLP constraints.

The quantification rules for IC follow those for expectations: if a variable occurs both in the $Body$ and in the $Head$, it is universally quantified with scope the whole IC. Otherwise, if it occurs in the $Head$, it is existentially quantified if it occurs in at least an **E** literal, and is universally quantified if it occurs only in **EN** literals. Function symbols and arbitrary nesting of terms are allowed; however we assume that the symbols **E**, **EN**, and **H** are used, in the user program, only as atom symbols, and not as term functors. We also assume that default negation is not applied to abducibles, including expectations.

Thanks to their implication structure and the deontic reading of expectations shown in [15], ICs can be read as conditional norms [24].

2.2. Case Study

In order to model article 96 ("Fraud or duress" from the Japanese Civil Code), we describe the content of events and expectations by means of the following terms:

- $intention(A, B, I, Id_I)$: person A utters a manifestation of intention to person B, with identifier Id_I for action I;
- $do(A, B)$: person A performs act B;
- $do(A, B, I)$: person A performs act B in the context of a contract identified by I;
- $induce(A, B)$: act A induces act B;
- $rescind(A, B, I, F, Id_I, Id_R)$: person A rescinds, with identifier Id_R, his or her intention, uttered to B and identified by Id_I, to perform action I, due to fraud or duress F;
- $know(A, F)$: person A becomes aware of fact F;
- $assertAgainst(A, B, Id_R)$: person A asserts the legal act identified by Id_R against person B.

Legally relevant acts ($intention$, $rescind$, $assertAgainst$) have identifiers.

The following integrity constraint states that a manifestation of intention should, in general, be followed by the performance of the act.

$$\mathbf{H}(intention(A, B, I, IdI), T_1) \rightarrow \mathbf{E}(do(A, I), T_2) \wedge T_2 > T_1 \tag{1}$$

Each of the following integrity constraints models one of the paragraphs of Article 96. Here, the predicate $fraudOrDuress/1$, defined in KB, specifies which actions count as fraud or duress.

1. Manifestation of intention which is induced by any fraud or duress may be rescinded.

$$\begin{aligned} &\mathbf{H}(intention(A, B, I, Id_I), T_1) \wedge \mathbf{H}(do(B, F), T_3) \wedge \mathbf{H}(induce(F, I), T_2) \\ &\wedge fraudOrDuress(F) \wedge T_3 < T_2 \wedge T_2 < T_1 \wedge T_1 < T_4 \\ \rightarrow &\neg \mathbf{EN}(rescind(A, B, I, F, Id_I, Id_R), T_4) \end{aligned} \tag{2}$$

2. In cases any third party commits any fraud or duress inducing any person to make a manifestation of intention to the other party, such manifestation of intention may be rescinded only if the other party knew such fact.

$$\begin{aligned} &\mathbf{H}(intention(A, B, I, Id_I), T_1) \wedge \mathbf{H}(do(C, F), T_3) \wedge C \neq B \\ &\wedge \mathbf{H}(know(B, F), T_5) \wedge \mathbf{H}(induce(F, I), T_2) \\ &\wedge fraudOrDuress(F) \wedge T_2 < T_1 \wedge T_3 \leq T_5 \wedge T_5 < T_1 \\ \rightarrow &\neg \mathbf{EN}(rescind(A, B, I, F, Id_I, Id_R), T_4) \wedge T_4 > T_1 \end{aligned} \tag{3}$$

3. The rescission of the manifestation of intention induced by the fraud or duress pursuant to the provision of the preceding two paragraphs may not be asserted against a third party without knowledge.

$$\begin{aligned} &\mathbf{H}(rescind(A, B, I, F, Id_I, Id_R), T_1) \wedge \mathbf{not}\ \mathbf{H}(know(C, F), T_2) \\ \rightarrow &\mathbf{EN}(assertAgainst(A, C, Id_R), T_3) \wedge T_1 < T_3 \end{aligned} \tag{4}$$

2.3. Declarative Semantics

The abductive semantics of the $\mathcal{S}$CIFF language defines, given a set $\mathbf{HAP}$ of $\mathbf{H}$ atoms called *history* and representing the actual behaviour, an abductive answer, i.e., a ground set $\mathbf{EXP}$ of expectations that:

- together with the history and KB, entails $\mathcal{IC}$

$$KB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathcal{IC} \tag{5}$$

  where $\models$ is entailment according to the 3-valued completion semantics [32], i.e., we require that each IC be true in all three valued model of the completion [33] of $KB \cup \mathbf{HAP} \cup \mathbf{EXP}$.

- is consistent with respect to explicit negation

$$\begin{array}{c}\{\mathbf{E}(Content, Time), \neg\mathbf{E}(Content, Time)\} \not\subseteq \mathbf{EXP} \wedge \\ \wedge \{\mathbf{EN}(Content, Time), \neg\mathbf{EN}(Content, Time)\} \not\subseteq \mathbf{EXP}\end{array} \tag{6}$$

- is consistent with respect to the meaning of expectations

$$\{\mathbf{E}(Content, Time), \mathbf{EN}(Content, Time)\} \not\subseteq \mathbf{EXP} \tag{7}$$

- is fulfilled by the history, i.e.

$$\text{if } \mathbf{E}(Content, Time) \in \mathbf{EXP} \text{ then } \mathbf{H}(Content, Time) \in \mathbf{HAP} \text{ and} \tag{8}$$

$$\text{if } \mathbf{EN}(Content, Time) \in \mathbf{EXP} \text{ then } \mathbf{H}(Content, Time) \notin \mathbf{HAP} \tag{9}$$

2.4. Operational Semantics

Operationally, the $\mathcal{S}$CIFF abductive proof procedure finds an abductive answer if one exists, or detects that no one exists (see [14] for soundness and completeness statements), meaning that the history violates the $\mathcal{S}$CIFF program. We call the two cases *success* and *failure*, respectively. The $\mathcal{S}$CIFF proof-procedure is defined through a set of transitions, each rewriting one node of a proof tree into one or more nodes.

Each node is a set of formulas, namely the partially solved integrity constraints (in the initial node, the whole set of integrity constraints), the constraints in the constraint store (initially empty), the set $\mathbf{HAP}$ of happened events (initially empty, it will register new events as they are detected) and the abducibles (also initially empty).

The basic transitions of $\mathcal{S}$CIFF are inherited from the IFF [21], and they account for the core of abductive reasoning. Other transitions deal with CLP constraints, and are inherited from the CLP [22] transitions.

The following is the subset of $\mathcal{S}$CIFF transitions that are relevant for this work, in a proof-theory style notation. In the following list, a is an abducible atom while p and q represent atoms of either abducible or defined predicates.

- propagation

$$\frac{a(N) \qquad a(N'), B \rightarrow H}{N = N', B \rightarrow H}$$

- unfolding

$$\frac{p(N) \qquad p(N') \leftarrow B}{N = N', B} \qquad\qquad \frac{\begin{array}{ll} & p(N') \leftarrow B' \\ & p(N'') \leftarrow B'' \\ p(N) \rightarrow H & \cdots \\ & p(N^k) \leftarrow B^k \end{array}}{\begin{array}{l} N = N', B' \rightarrow H \\ N = N'', B'' \rightarrow H \\ \cdots \\ N = N^k, B^k \rightarrow H \end{array}}$$

- case analysis

$$\frac{(N = N', B) \rightarrow H}{N = N' \quad B \rightarrow H \qquad \vee \qquad N \neq N'}$$

- equality rewriting

$$\frac{[\exists E][\forall A] A = E}{\theta = \{A/E\}} \qquad\qquad \frac{[\exists E][\forall A] A \neq E}{false}$$

$$\frac{[\exists E_1][\exists E_2] E_1 = E_2}{\theta = \{E_1/E_2\}}$$

$$\frac{X = t \qquad t \text{ does not contain } X}{\theta = \{X/t\}}$$

$$\frac{X = t \qquad t \text{ contains } X}{false} \qquad\qquad \frac{X \neq t \qquad t \text{ contains } X}{true}$$

$$\frac{p(t_1, \ldots, t_n) = p(s_1, \ldots, s_n)}{t_1 = s1, \ldots, t_n = s_n} \qquad\qquad \frac{p(t_1, \ldots, t_n) \neq p(s_1, \ldots, s_n)}{t_1 \neq s1 \vee \cdots \vee t_n \neq s_n}$$

$$\frac{p(t_1, \ldots, t_n) = q(s_1, \ldots, s_m) \text{ where } p \neq q \vee n \neq m}{false}$$

$$\frac{p(t_1, \ldots, t_n) \neq q(s_1, \ldots, s_m) \text{ where } p \neq q \vee n \neq m}{true}$$

where the θ substitution is added to the child node. In case two variables with different quantifiers are unified, the new variable is existentially quantified.

- logical simplifications

$$\frac{true \rightarrow A}{A} \qquad \frac{false \rightarrow A}{true} \qquad \frac{true \wedge A}{A}$$

$$\frac{false \wedge A}{false} \qquad \frac{true \vee A}{true} \qquad \frac{false \vee A}{A}$$

In order to deal with the concept of expectation, abduced expectations can become *fulfilled*; an expectation $\mathbf{E}(X,T)$ (respectively, $\mathbf{EN}(X,T)$) is declared fulfilled using the notation $\mathbf{F}(\mathbf{E}(X,T))$ (respectively $\mathbf{F}(\mathbf{EN}(X,T))$). We call $\mathbf{FULF} = \{\mathbf{E}(X,T)|\mathbf{F}(\mathbf{E}(X,T))\} \cup \{\mathbf{EN}(X,T) \mid \mathbf{F}(\mathbf{EN}(X,T))\}$ the set of fulfilled expectations in a node, and $\mathbf{PEND} = \mathbf{EXP} \setminus \mathbf{FULF}$ the set of expectations that are not fulfilled.

Transition *Fulfillment* $\mathbf{E}$ deals with the fulfillment of $\mathbf{E}$ expectations: if an expectation $\mathbf{E}(E,T_E) \in \mathbf{PEND}$ and the event $\mathbf{H}(H,T_H)$ is in the current history $\mathbf{HAP}$, two nodes are generated: one in which $E = H, T_E = T_H$ and $\mathbf{E}(E,T_E)$ is fulfilled, the other in which $E \neq H$ or $T_E \neq T_H$.

$$\frac{\mathbf{E}(E,T_E), \mathbf{H}(H,T_H)}{(E = H, T_E = T_H, \mathbf{F}(\mathbf{E}(E,T_E))) \ \vee \ E \neq H \vee T_E \neq T_H}$$

Transition *Violation* $\mathbf{EN}$ deals with the violation of $\mathbf{EN}$ expectations: if an expectation $\mathbf{EN}(E,T_E) \in \mathbf{PEND}$ and the event $\mathbf{H}(H,T_H) \in \mathbf{HAP}$, one node is generated where the constraint $E \neq H \vee T_E \neq T_H$ is imposed, possibly leading to failure.

$$\frac{\mathbf{EN}(E,T_E), \mathbf{H}(H,T_H)}{E \neq H \vee T_E \neq T_H}.$$

When there are no more relevant events, *history closure* is applied, and the history is declared closed. In this case, transition *Violation* $\mathbf{E}$ becomes applicable: all $\mathbf{E}$ expectations in $\mathbf{PEND}$ are considered as violated and failure occurs.

$$\frac{history\_closed, \mathbf{E}(E,T_E), \nexists \mathbf{F}(\mathbf{E}(E,T_E))}{false}$$

As regards complexity, the $\mathcal{S}$CIFF language is an extension of Prolog, and, as such, it is Turing-complete; so a $\mathcal{S}$CIFF evaluation, in general, may not terminate. Even in the propositional case, Gottlob and Eiter [34] proved that the complexity of abduction is Σ_2^P-complete.

Example 2.1. We now show a simple example of a one-branch $\mathcal{S}$CIFF derivation.

Let a KB be composed of the clause

$$p(a, 10)$$

and the integrity constraint

$$\mathbf{H}(a,T_1) \wedge p(a,T) \rightarrow \mathbf{E}(b,T_2) \wedge T_2 < T_1 + T$$

The initial node has the IC as the only partially solved integrity constraint. Let the event $\mathbf{H}(a, 5)$ be detected. By *propagation*, a child node with the partially solved IC $p(a, T) \rightarrow \mathbf{E}(b, T_2) \wedge T_2 < 5 + T$ is created, and subsequently (by *unfolding* and *case analysis*) a node containing expectation $\mathbf{E}(b, T_2)$ and a constraint $T_2 < 15$. Then, let the event source terminate, so $history\_closed$ is applied.

Since expectation $\mathbf{E}(b, T_2)$ is not fulfilled and the history is closed, violation is applied to $\mathbf{E}(b, T_2)$.

3. $\mathcal{S}$CIFF$^{\mathcal{D}}$ Language

In legal reasoning, expectations can be discharged not only because they become fulfilled by matching the actual behaviour of the agent, but also for other reasons. For example, in case a contract is declared null, the agents are no longer expected to perform the actions required in the contract.

We introduce an extension of the $\mathcal{S}$CIFF language to deal with expectations that do not hold any longer. We introduce a new abducible atom, $\mathbf{D}(\mathbf{E})$, that means that an expectation is discharged; the atom

$$\mathbf{D}(\mathbf{E}(X, T))$$

means that the expectation $\mathbf{E}(X, T)$ is not required to be fulfilled by an event, as it has been discharged.

The integrity constraints can have $\mathbf{D}$ atoms, that can be abduced; in $\mathcal{S}$CIFF$^{\mathcal{D}}$, we let the symbol $\mathbf{E}$ occur in a term only within a $\mathbf{D}(\mathbf{E}(X, T))$ atom. More precisely, if $t(\mathbf{E}(X, T))$ is an atom or term containing an expectation, then the functor symbol t must be $\mathbf{D}$.

Note that $\mathbf{D}$ atoms have only the discharged expectation as argument, i.e., expectation dischargement does not have a time associated to it. This choice allows us to ensure the formal properties of the $\mathcal{S}$CIFF$^{\mathcal{D}}$ proof procedure (see Section 4.3).

Example 3.1. *The user might write an IC saying that, if a contract with identifier Id_C is null, all expectations requiring an action in the context of that contract are discharged. To express that a contract is null, we introduce a binary abducible* $\mathbf{NULL}$*, carrying the identifier of the contract and that of the reason for nullification as arguments.*

$$\begin{aligned} \mathbf{NULL}(Id_C, Id_{null}) \wedge \mathbf{E}(do(Agent, Action, Id_C), T_{do}) \\ \rightarrow \mathbf{D}(\mathbf{E}(do(Agent, Action, Id_C), T_{do})). \end{aligned} \tag{10}$$

We can express that a contract that is explicitly permitted to be rescinded can be nullified as

$$\begin{aligned} \neg\mathbf{EN}(rescind(A, I, F, Id_I, Id_R), T_r) \wedge \mathbf{H}(rescind(A, I, F, Id_I, Id_R), T_r) \\ \rightarrow \mathbf{NULL}(Id_I, Id_R) \end{aligned} \tag{11}$$

The combined effect of ICs (10) and (11) is that rescission is only effective when the circumstances grant an explicit permission.

Example 3.2. *As a second case study from the Japanese Civil Code, we consider Article 130, which states:* "In cases any party who will suffer any detriment as a result of the fulfillment of a condition

intentionally prevents the fulfillment of such condition, the counterparty may deem that such condition has been fulfilled". *Article 130 can be modeled as follows:*

$$\begin{aligned}&\mathbf{H}(do(Agent_1, Action_1), T_1) \wedge \mathbf{E}(do(Agent_2, Action_2), T_2)\\ &\wedge detrimental(Action_2, Agent_1) \wedge prevent(Action_1, Action_2)\\ &\rightarrow \mathbf{D}(\mathbf{E}(do(Agent_2, Action_2), T_2))\end{aligned} \tag{12}$$

where detrimental/2 and prevent/2 are predicates defined in the KB to specify when, respectively, an action is detrimental to an agent and when an action prevents another.

In legal reasoning it is also necessary to provide means to nullify the nullification of a contract; in our running example even the rescission of a contract could be due to some illegal action, such as duress. In order to re-state expectations for contracts that were wrongly nullified, we provide the following definition.

Definition 3.3. (Reinstatement)
A discharged expectation $\mathbf{D}(\mathbf{E}(X, T))$ can be *reinstated*, with the syntax

$$\mathbf{E}(\mathbf{D}(\mathbf{E}(X, T))).$$

Note that, to reinstate a discharged expectation $\mathbf{E}(X, T)$, it would not be sufficient to simply state it again. The set of expectations is monotonic, so $\mathbf{E}(X, T)$ is still in the set of expectations when it is discharged by adding $\mathbf{D}(\mathbf{E}(X, T))$. By only adding $\mathbf{E}(X, T)$ again, the set containing $\mathbf{E}(X, T)$ and $\mathbf{D}(\mathbf{E}(X, T))$ would be unchanged, and $\mathbf{E}(X, T)$ would still be discharged. $\mathbf{E}(\mathbf{D}(\mathbf{E}(X, T)))$ in the set of expectations shows that the discharged expectation $\mathbf{E}(X, T)$ is reinstated.

Example 3.4. *The following integrity constraint models the fact that an expectation that was discharged because of the nullified nullification of a contract is reinstated.*

$$\begin{aligned}&\mathbf{NULL}(\mathbf{NULL}(IdContract, IdRescission), T_r)\\ &\wedge\mathbf{D}(\mathbf{E}(do(A, Action, IdContract), T_{exp}))\\ &\rightarrow\mathbf{E}(\mathbf{D}(\mathbf{E}(do(A, Action, IdContract), T_{exp}))).\end{aligned} \tag{13}$$

The unary abducible $\mathbf{NULL}$*, when applied to a* $\mathbf{NULL}$ *term, means that the nullification is itself nullified.*

Syntactic restrictions The following syntactic restriction is used in the proof of soundness.

Definition 3.5. (Weak D-allowedness)
An IC containing a $\mathbf{D}$ atom is *weakly* $\mathbf{D}$*-allowed* if there is only one $\mathbf{D}$ atom, it occurs in the head, and the head contains only that atom. A KB is weakly $\mathbf{D}$-allowed if none of its clauses contains $\mathbf{D}$ atoms. A $\mathcal{S}$CIFF$^{\mathcal{D}}$ program $\langle KB, \mathcal{IC}\rangle$ is weakly $\mathbf{D}$-allowed if KB is weakly $\mathbf{D}$-allowed and all the ICs in $\mathcal{IC}$ are weakly $\mathbf{D}$-allowed.

The following restriction is not necessary for the soundness results proved in Sect. 4.3, but it allows a more efficient treatment of the $\mathbf{D}$ atoms. Note that all the examples presented in this paper satisfy the restriction.

Definition 3.6. (Strong D-allowedness)
An IC containing a $\mathbf{D}$ atom is *strongly* $\mathbf{D}$*-allowed* if it is weakly $\mathbf{D}$-allowed and the (only) expectation in the $\mathbf{D}$ atom occurs identically in the body of the IC.

Intuitively, the given notion of strong $\mathbf{D}$-allowedness lets one define ICs that select one expectation and make it discharged, possibly subject to further conditions occurring in the body. This syntactic restriction is aimed at capturing the most common scenarios while trying to maintain an efficient execution.

In fact, if the strong allowedness condition is lifted, it is not required for an atom $\mathbf{E}(X,T)$ to have been abduced before declaring it discharged. If two atoms $\mathbf{E}(X,T)$ and $\mathbf{D}(\mathbf{E}(Y,U))$ are abduced, two options have to be explored, as alternatives: either (X,T) unifies with (Y,U), and the expectation $\mathbf{E}(X,T)$ becomes discharged, or (X,T) and (Y,U) do not unify (e.g., by imposing a dis-unification constraint $(X,T) \neq (Y,U)$). In these cases, the $\mathcal{S}$CIFF proof-procedure opens a choice point; this means that, in case $|E|$ expectations $\mathbf{E}$ are abduced and $|D|$ $\mathbf{D}$ atoms are abduced, $|E||D|$ choice points will be created, each opening 2 alternative branches, which would generate $2^{|E||D|}$ branches.

We performed an experiment to verify this worst-case analysis. We generated a number of $\mathbf{E}(X,T)$ and $\mathbf{D}(\mathbf{E}(Y,U))$ atoms, and measured the time $\mathcal{S}$CIFF$^{\mathcal{D}}$ took to find the first solution and all solutions (all experiments were run on a Intel Core i7-3720QM CPU @ 2.60GHz running SWI-Prolog version 7.4.0-rc1 on Linux Mint 18.1 Serena 64 bits). For all solutions (Figure 1 right), the running time follows closely the foreseen $2^{|E||D|}$, while for one solution (Figure 1 left), the running time seems dependent mainly on the number of raised expectations and almost independent from the number of $\mathbf{D}$ atoms. Note also the different scales: finding one solution takes at most 3 seconds with 100 expectations and discharge atoms, while finding all solutions takes almost 3 hours with $|E| = |D| = 8$.

From a language viewpoint, the strong allowedness condition restricts the set of expectations that can be discharged to those that have been raised. Without such restriction, one could abduce a generic atom $\mathbf{D}(\mathbf{E}(X,T))$ saying that one expectation is discharged. Semantically, this would mean that one of the expectations might be discharged, although it is not said which one.

Dischargement scenarios The remainder of this section is devoted to the discussion of an example concerning manifestation of intention (Section II of the Japanese Civil Code), modeled in $\mathcal{S}$CIFF$^{\mathcal{D}}$, and one concerning prevention of fulfillment of conditions. Sections presenting the declarative and operational semantics of the extended framework and proof of soundness then follow.

Example 3.7. *(Example 3.1 continued).*

Let us consider the case study of section 2.2 again, and assume that work on an acquired good G is to be paid by the good's owner O, unless O rescinds the good's purchase and asserts the rescission against the performer of the work M (which, due to integrity constraint (4), is only allowed if the

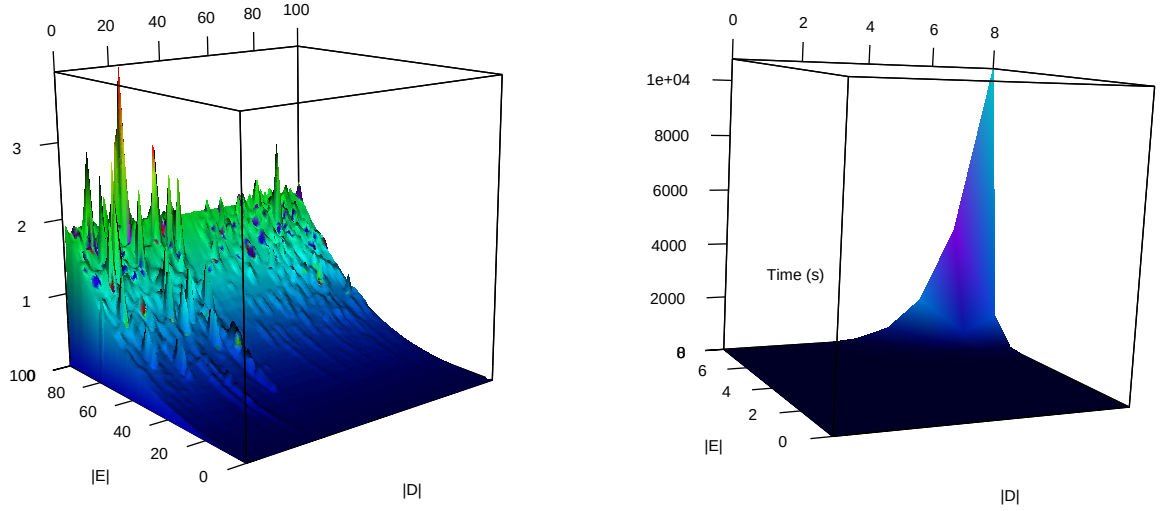

Figure 1. Experiments with different numbers $|D|$ of **D** atoms and $|E|$ of **E** atoms; time in seconds for finding one solution (left) or all solutions (right).

performer was aware of the rescission's cause, such as a fraud). We can express this norm by means of the following integrity constraint:

$$\begin{aligned} &\mathbf{H}(work(M,G,O,W),T_1)\\ \rightarrow&\mathbf{E}(pay(O,M,W),T_2) \wedge T_1 < T_2\\ \vee(&\mathbf{E}(rescind(O,B,buy(G),F,Id_I,Id_R),T_3)\\ &\wedge \mathbf{E}(assertAgainst(O,M,Id_R),T_4)\\ &\wedge T_1 < T_3 \wedge T_3 < T_4) \end{aligned} \tag{14}$$

where the term $work(M,G,O,W)$ represents mechanic M doing work W on the good G owned by O, and the term $pay(O,M,W)$ represents owner O paying mechanic M for work W.

The KB states that fixing a car's mileage constitutes fraud or duress.

$$fraudOrDuress(fixMileage(C)) \tag{15}$$

In the scenario defined by the following history

$$\begin{aligned} &\mathbf{H}(do(bob, fixMileage(car)), 1)\\ &\mathbf{H}(induce(fixMileage(car), buy(car)), 2)\\ &\mathbf{H}(intention(alice, bob, buy(car), i1), 3)\\ &\mathbf{H}(work(mechanic, car, alice, lpgSystem), 4)\\ &\mathbf{H}(rescind(alice, bob, buy(car), fixMileage(car), i1, i2), 5) \end{aligned} \tag{16}$$

Alice's rescission is explicitly permitted by IC (2), because her manifestation of intention was induced by Bob's fraudulent act of fixing the car's mileage; the expectation $\mathbf{E}(do(alice, buy(car)), T)$*, raised because of IC (1), is discharged because of ICs (10) and (11).*

However, since the mechanic was not aware of Bob's fraud, IC (4) prevents Alice from asserting the rescission against him, so the second disjunct in IC (14) cannot hold, and Alice still has to pay him for installing the LPG system ($\mathbf{E}(pay(alice, mechanic, lpgSystem, T_2))$*). For the history that contains all the events in formula (16), plus* $\mathbf{H}(know(mechanic, fixMileage(car)), 1)$ *(i.e., the mechanic is now aware of the car's mileage being fixed),* $alice$ *is not prohibited from asserting the rescission against the mechanic by integrity constraint (4). With the event* $\mathbf{H}(assertAgainst(alice, mechanic, i2), 6)$*, the second disjunct in the head of integrity constraint (14) is satisfied and* $alice$ *is not obliged to pay the mechanic for his work.*

Example 3.8. *Consider the following scenario, where an order by a customer should be followed by a delivery by the seller:*

$$\begin{aligned} &\mathbf{H}(order(Customer, Seller, Good), T_{order}) \\ &\rightarrow \mathbf{E}(do(Seller, deliver(Good)), T_{delivery}) \wedge T_{delivery} > T_{order}. \end{aligned} \tag{17}$$

Suppose that Alice placed an order, but in the meanwhile she was diagnosed a rare immunodeficiency, and she cannot meet people, except her family members. Her mother usually lives with her, but today she went out, so Alice locked the door, as it would be detrimental for her if any person got in the house. This prevents any delivery, but it is a minor issue for her compared to the consequences that Alice should face in case she met a stranger.

$$\begin{aligned} &detrimental(deliver(Good), alice). \\ &prevent(lockDoor, deliver(Good)). \end{aligned} \tag{18}$$

Given the following history

$$\begin{aligned} &\mathbf{H}(order(alice, bob, computer), 1) \\ &\mathbf{H}(do(alice, lockDoor), 2) \end{aligned} \tag{19}$$

the expectation for Bob to deliver the good, raised by IC (17), is discharged by IC (12), because Alice performed an action that prevents the fulfillment.

Example 3.9. *The following IC states that if an agent* A *expresses an intention to buy an item for sale from agent* B*, then* A *should pay* B *for the item, and* B *should sell it.*

$$\begin{aligned} &\mathbf{H}(intention(A, B, buy(Item), IdI), T_i) \wedge forSale(Item) \\ &\rightarrow \mathbf{E}(do(A, pay(B, Item), IdI), T_p) \\ &\wedge \mathbf{E}(do(B, sell(A, Item), IdI), T_s) \end{aligned} \tag{20}$$

Consider IC (20) together with ICs (2), (10) and (11), and the following KB *that states that a threat is a kind of fraud or duress and that a piece of land is for sale:*

$$\begin{array}{l} fraudOrDuress(threat) \\ forSale(land) \end{array} \tag{21}$$

Now, alice bought a piece of land for sale from bob, but later rescinded the purchase stating that she had committed to it due to clyde's duress, as described by the following history:

$$\begin{array}{l} \mathbf{H}(do(clyde, threat), 1) \\ \mathbf{H}(induce(threat, buy(land)), 2) \\ \mathbf{H}(intention(alice, bob, buy(land), id_b), 3) \\ \mathbf{H}(rescind(alice, bob, buy(land), threat, id_b, id_r), 5) \end{array} \tag{22}$$

The expectations for alice to pay bob and for bob to sell to alice are therefore discharged.

Now consider a plot twist in this story: it was later found out that alice decided to rescind the contract due to bob's duress: after the signature of the contract, dale proposed a higher price to bob, who was now bound to sell to alice. So, bob decided to threaten alice who, in fear of bob's unlawful behavior, found an excuse to rescind the contract, although the rescission was not in her economic interest.

$$\begin{array}{l} \mathbf{H}(do(bob, threat), 3) \\ \mathbf{H}(induce(threat, rescind(alice, bob, buy(land), threat, id_b, id_r)), 4) \end{array} \tag{23}$$

It is reasonable to expect alice's rescission to be ineffective, since it was induced by duress. This can be achieved by imposing the following IC together with IC (13):

$$\begin{array}{l} \quad \mathbf{H}(rescind(A, B, I, F, IdC, IdR), T_r) \\ \wedge \mathbf{H}(do(B, F), T_f) \\ \wedge \mathbf{H}(induce(F, rescind(A, B, I, F, IdC, IdR)), T_i) \\ \quad fraudOrDuress(F) \wedge T_f < T_i < T_r \\ \wedge \mathbf{NULL}(IdC, IdR) \\ \rightarrow \mathbf{NULL}(\mathbf{NULL}(IdC, IdR)) \end{array} \tag{24}$$

Now, thanks to the reinstatement, when bob's duress is found, all the expectations related to the purchase contract (e.g., that alice has to pay the sum that was agreed upon, and that bob has to sell the estate to alice) are reinstated.

4. $\mathcal{S}$CIFF$^{\mathcal{D}}$ Declarative and Operational Semantics

4.1. Declarative Semantics

We now show how to deal with the discharge of expectations in the context of ALP. We first give an intuitive definition, then show its pitfalls and finally provide a correct definition.

In order to accept histories in which expectations may not have matching events, we need to extend the definition of fulfillment of expectations given in equation (8); intuitively, a positive expectation is fulfilled if either there is a matching event or if the expectation has been discharged:

$$\begin{array}{rl} \text{if} & \mathbf{E}(Content, Time) \in \mathbf{EXP} \\ \text{then} & \mathbf{H}(Content, Time) \in \mathbf{HAP} \vee \mathbf{D}(\mathbf{E}(Content, Time)) \in \mathbf{EXP} \end{array} \tag{25}$$

We redefine the concept of fulfillment, considering that expectations could be reinstated, with any nesting level.

Definition 4.1. (Fulfilment in case of reinstatement)
An expectation $\mathbf{E}(C_E, T) \in \mathbf{EXP}$ is *fulfilled* if

$$\begin{array}{c} (\mathbf{H}(C_H, T) \in \mathbf{HAP} \wedge match(C_H, C_E, T)) \\ \vee \mathbf{D}(\mathbf{E}(C_E, T)), \end{array} \tag{26}$$

where $match$ is recursively defined as

$$match(C_H, C_E, T) = \begin{cases} True & if\ C_H = C_E \\ & \quad \vee\ C_E = \mathbf{D}(\mathbf{E}(C'_E, T)) \wedge match(C_H, C'_E, T) \\ False & otherwise. \end{cases} \tag{27}$$

$match$ allows to match events with reinstated expectations. It is defined recursively: the content C_H of an event matches the content C_E of an expectation at time T if they are equal, or C_E is reinstatement of an expectation of content C'_E that matches C_H at time T.
The introduction of the disjunction in equation (25) (and, respectively, eq. (26) for the case of reinstatement) opens alternative ways of fulfilling expectations. However, in this way there could exist abductive answers with $\mathbf{D}$ literals even if there is no explicit rule introducing them. For example, in the history of formula (16) an abductive answer would be[1] (where each underscore represents an unnamed variable, as in Prolog)

$$\begin{array}{l} \neg\mathbf{EN}(rescind(alice, bob, buy(car), fixMileage(car), i1, \_), T2), \\ \mathbf{D}(\mathbf{E}(do(alice, buy(car), i1), \_)) \\ \mathbf{D}(\mathbf{E}(pay(alice, mechanic, lpgsystem), \_)) \\ \mathbf{NULL}(i1, i2) \end{array}$$

since it satisfies equations (5) (which takes into account knowledge base and integrity constraints), (6), (7), (9) and (25). Note that Alice is no longer required to pay the mechanic, because although no IC introduces explicitly the dischargement of the expectation that she should pay, the abductive semantics accepts the introduction of the literal $\mathbf{D}(\mathbf{E}(pay(alice, mechanic, lpgsystem), \_))$.

In order to avoid the abduction of $\mathbf{D}$ atoms that were not explicitly introduced by a rule, we use subset minimality of the set of $\mathbf{D}$ atoms.

[1]For brevity, we omit an expectation $\mathbf{E}(x)$ if we have already its discharged version $\mathbf{D}(\mathbf{E}(x))$.

Definition 4.2. (Abductive answer)
If there is a set $\mathbf{EXP}$ that

1. satisfies equations (5), (6), and (7)
2. is minimal with respect to set inclusion within the sets satisfying point 1, considering only $\mathbf{D}$ atoms; more precisely: there is no set $\mathbf{EXP'}$ satisfying point 1 and such that $\mathbf{EXP'} \subset \mathbf{EXP}$ and $\mathbf{EXP} = \mathbf{EXP'} \cup F$, where F contains only $\mathbf{D}$ atoms
3. satisfies equations (9) and (26)

then the set $\mathbf{EXP}$ is an abductive answer, and we write $\langle KB, \mathcal{IC} \rangle \models_{\mathbf{EXP}} true$.

4.2. Operational Semantics

Operationally, transition *Violation* $\mathbf{E}$ is updated as follows:

$$\frac{history\_closed, \mathbf{E}(E, T_E), \nexists \mathbf{F}(\mathbf{E}(E, T_E)), \nexists \mathbf{D}(\mathbf{E}(E, T_E))}{false} \tag{28}$$

i.e., a positive expectation is violated if it is neither fulfilled nor discharged.

In order to deal with the reinstatement of expectations, we also extend transition *Fulfillment* $\mathbf{E}$. We distinguish two cases: one in which the expectation is not a reinstatement:

$$\frac{\mathbf{E}(E, T_E), \mathbf{H}(H, T_H), E \neq \mathbf{D}(\mathbf{E}(\_, \_))}{(E = H, T_E = T_H, \mathbf{F}(\mathbf{E}(E, T_E))) \ \vee \ E \neq H \vee T_E \neq T_H}$$

(where each underscore represents an unnamed variable, as in Prolog) and one that deals with reinstatement:

$$\frac{\mathbf{E}(\mathbf{D}(\mathbf{E}(C_E, T_E)), T_E), \mathbf{H}(H, T_H), E = inner(C_E)}{(E = H, T_E = T_H, \mathbf{F}(\mathbf{E}(\mathbf{D}(\mathbf{E}(C_E, T_E), T_E)))) \ \vee \ E \neq H \vee T_E \neq T_H} \tag{29}$$

where $inner$ is recursively defined as

$$inner(C) = \begin{cases} inner(X) & \text{if } C = \mathbf{D}(\mathbf{E}(X, T_X)) \\ C & otherwise. \end{cases} \tag{30}$$

Note that, since the content C of an expectation $\mathbf{E}(C, T)$ cannot be a variable (Section 2.1), it is possible, at evaluation time, to distinguish which of the two alternatives is to be taken. For this reason, the *otherwise* branch in Equation (30) does not require imposing a dis-equality constraint $C \neq \mathbf{D}(\mathbf{E}(X, T_X))$.

If a computation terminates with success, we write $\langle KB, \mathcal{IC} \rangle \vdash_{\mathbf{EXP}} true$.

4.3. Soundness

We are now ready to give the soundness statements; these statements rely on the soundness and completeness theorems of the $\mathcal{S}$CIFF proof-procedure, so they hold in the same cases; for the $\mathcal{S}$CIFF allowedness conditions over knowledge base and integrity constraints, we refer the reader to [14].

Theorem 4.3. (Soundness of success) If a $\mathcal{S}$CIFF$^{\mathcal{D}}$ program $\langle KB, \mathcal{IC}\rangle$ is weakly $\mathbf{D}$-allowed and $\langle KB, \mathcal{IC}\rangle \vdash_{\mathbf{EXP}} true$ then $\langle KB, \mathcal{IC}\rangle \models_{\mathbf{EXP}} true$.

Proof:
If no $\mathbf{D}$ atoms occur in $\mathcal{IC}$, the procedure coincides with $\mathcal{S}$CIFF, which is sound [14]. In the case with $\mathbf{D}$ atoms in $\mathcal{IC}$, the $\mathcal{S}$CIFF$^{\mathcal{D}}$ procedure might report success in cases in which $\mathcal{S}$CIFF reports failure due to the extended notion of fulfillment (eq. (26)), mapped in the updated transitions reported in Section 4.2. An expectation $\mathbf{E}(C_E, T)$ could cause failure in $\mathcal{S}$CIFF and not in $\mathcal{S}$CIFF$^{\mathcal{D}}$ because

1. the expectation was discharged: $\mathbf{D}(\mathbf{E}(C_E, T))$, and the new version of transition *violation* $\mathbf{E}$ (Eq. (28)) is no longer applicable
2. there exists an event $\mathbf{H}(C_H, T)$, where $C_H \neq C_E$ but $inner(C_E) = C_H$ and the new version of transition *Fulfillment* $\mathbf{E}$ (Eq. (29)) is now applicable.

In the first case, the $\mathbf{D}$ atom must have been generated, and the only way to generate it is through an IC having such atom in the head (see Definition 3.5).

The procedure generates the atom only if the body of the IC is true. If the body is true, it means (from the soundness of $\mathcal{S}$CIFF) that it is true also in the declarative semantics, so the $\mathbf{D}$ atom must be true also declaratively. In such a case, eq. (26) is satisfied, meaning that the success was sound.

In the second case, clearly $match(C_H, C_E, T)$ is true, so equation (26) is satisfied. □

Theorem 4.4. (Soundness of failure) If a $\mathcal{S}$CIFF$^{\mathcal{D}}$ program $\langle KB, \mathcal{IC}\rangle$ is weakly $\mathbf{D}$-allowed and $\langle KB, \mathcal{IC}\rangle \models_{\mathbf{EXP}} true$, then $\exists \mathbf{EXP'} \subseteq \mathbf{EXP}$ such that $\langle KB, \mathcal{IC}\rangle \vdash_{\mathbf{EXP'}} true$

Proof:
If there are no $\mathbf{D}$ atoms in $\mathcal{IC}$, the procedure coincides with $\mathcal{S}$CIFF, so the completeness theorem of $\mathcal{S}$CIFF holds [14]. In the case with $\mathbf{D}$ atoms in $\mathcal{IC}$, the declarative semantics (Eq. (26)) allows as abductive answers some sets that would not have been returned by $\mathcal{S}$CIFF, and in which some expectation $\mathbf{E}(C_E, T)$

1. either is not fulfilled by an actual event, but discharged through an abduced $\mathbf{D}$ atom
2. or it is fulfilled by an event $\mathbf{H}(C_H, T)$ such that $C_E \neq C_H$, but $match(C_H, C_E, T)$.

In the first case, consider an abductive answer $\mathbf{EXP}$ containing at least an expectation matching a $\mathbf{D}$ atom. We prove by contradiction that each $\mathbf{D}$ atom in $\mathbf{EXP}$ occurs in the head of an IC whose body is true. In fact, if a $\mathbf{D}$ atom in $\mathbf{EXP}$ was not in the head of an IC whose body is true, then the set $\mathbf{EXP''}$ obtained by removing the $\mathbf{D}$ atom from $\mathbf{EXP}$ would satisfy equation (5). On the other hand, since $\mathbf{EXP}$ satisfies equations (6) and (7) and those equations do not involve $\mathbf{D}$ atoms, also $\mathbf{EXP''}$

satisfies those equations. This means that $\mathbf{EXP}$ does not satisfy condition 2 of Definition 4.2, which means that $\mathbf{EXP}$ was not an abductive answer and we get a contradiction.

Since each $\mathbf{D}$ atom occurs in the head of an IC whose body is true, the procedure applies such IC and abduces that atom. This means that the corresponding expectation becomes discharged, and hence it does not cause failure.

In the second case, clearly $inner(C_E) = C_H$, meaning that transition *Fulfillment* $\mathbf{E}$ (Eq (29)) is applicable, and the expectation $\mathbf{E}(C_E, T)$ becomes fulfilled, so it will not cause failure when the history becomes closed (Eq. (28)). □

5. Related Work

Many authors have applied computational logic techniques (logic programming and extensions, answer set programming) to normative reasoning. In the following, we review some of those approaches that support some kind of defeasible obligation, related to our notion of dischargeable expectation.

Ryu and Lee [35] provide a first-order framework of deontic reasoning that can model and compute social regulations and norms. They employ defeasible reasoning in order to represent and manage counterfactual implications. In their framework, deontic operators are represented as first order terms; a specification is given as a set of strict and defeasible clauses. The operational semantics of their language consists of a SLD resolution-based computation process. The main purpose of our work is similar to that of Ryu and Lee's work: to give a computational method for systems specified by means of deontic operators. The works are also similar in the representation of deontic operators (as first order terms). However, our work is based on abduction, rather than on defeasible logic.

Satoh et al.'s PROLEG is a Prolog implementation of the Presupposed Ultimate Fact Theory of the Japanese Civil Code [6].

Contract cancellation under the Japanese law was also formalized in computational logics in the work by De Vos et al. [7], which models legal status of contracts in terms of institutions. They use a representation of legal status *à la* event calculus [36] to represent initiation and termination of validity of contracts; then the status is translated into answer set programming [37] to compute the current legal status. However, it inherits a drawback of answer set programming, that is, they must ground each variable into constant symbols and it results in the fact that all the status must be grounded with every time stamp, which is computationally expensive. On the other hand, our work is based on an extension of abductive logic programming so that we can use variables and unifications to avoid unnecessary grounding.

A notable application of Abductive Logic Programming to normative reasoning is Kowalski and Satoh's Normative ALP Frameworks [38]. Normative ALP Frameworks are abductive logic programs with a preference relation defined as a partial order over models; a formula is obligatory in a Normative ALP if it is true in all its optimal (i.e., such that no other model is preferred) models. The authors show that Normative ALP frameworks provide elegant solutions to several normative system paradoxes. While being based on abductive logic programming like the present work, their approach to normative system specification differs from ours in two significant ways: first, it requires a partial order to be defined over the set of models; second, obligation is not expressed in a specification by the normative system designer, rather it is an implied property of the set of models.

We now review approaches to defeasibility from outside the field of logic programming. While they provide elegant and expressive models, integrating them into a framework like $\mathcal{S}$CIFF while keeping its correctness and computational properties presents significant challenges.

Input/Output logic [39] is a framework for reasoning about conditional norms. A conditional norm is an implication written as (a, x) [40] and stating that if a is the case, then x is obligatory. Typically, both a and x are taken to be formulae from propositional logic [41]. Input/Output logic is based on the idea of detachment, that itself is based on modus ponens. The difference with modus ponens is that when the antecedent a is true, then the head x is detached, and the set of detached consequences is reasoned upon by subsequent steps in the semantics. Given a conditional rule, there are various ways in which the conclusion about what is obligatory can be detached, providing different logical systems, that can handle various issues in Deontic Logic, such as reasoning about norm violations or constitutive norms. In the $\mathcal{S}$CIFF framework, reasoning on norm violations is currently not possible; on the other hand $\mathcal{S}$CIFF can reason upon non-propositional terms.

Boella and Van der Torre [42] present an approach based on input/output logics [39] for formalizing conditional norms, obligations and permissions in a scenario where many hierarchically organized authorities are present. In such a scenario, there can be norms that are more important than others and therefore the authors consider a hierarchy of norms, defined by "meta-norms", and different types of permissions with different strengths. However, the focus of [42] is on helping the legislator to understand how the modification of a norm or the definition of a new one may change the whole normative system. In fact, following the input/output logic's semantics, [42] is not concerned about the truth value of formulae representing (part of) norms but defines a cause-effect link between inputs and obligatory outputs in an abductive-like way.

Governatori et al. [43] propose to model a rich set of fundamental concepts, such as right, noright, duty, and privilege, in a multi-modal computational framework. They exploit computational features of deontic logic and set specific rules for introducing modal operators, where rules are primarily meant to introduce modalities in terms of provability of literals. A literal will be modalised with, say, X if it is deduced via rules specifically devised to express concept X. They exploit normative conditionals, and dedicated rules for inferring, e.g., persistent obligations (in our terminology obligations that cannot be discharged) or co-occurrent obligations. These last rules allow for the inference of obligations which hold on the condition and only while the antecedents of these rules hold (aka of automatic dischargement).

The temporal evolution of a normative systems has been studied in several deontic logics, such as the following.

Governatori et al. [44] discuss the impact new contracts, which introduce new constraints, may have on already existing business processes. The authors present a logic called FCL (Formal Contract Language), based on RuleML, for representing contracts in a formal way. The language allows automatic checking and debugging, analysis and reasoning [45]. In [44] a normal form of FCL, called NFCL, is presented with the aim of having a clean, complete and non-redundant representation of a contract. This normal form is obtained by merging the new constraints with the existing ones and cleaning up the redundancies by using the notion of subsumption. The result points out possible conflicts among contracts and how each contract is intertwined with the whole business process. Similar results can be accomplished with $\mathcal{S}$CIFF$^{\mathcal{D}}$, which allows checking the consistency of the $\mathcal{S}$CIFF$^{\mathcal{D}}$

program representing the constraints of the business process.

The **AD** system [46] is a deontic logic that supports defeasibile obligations by means of a revision operator (called f), which represents the assumptions that normally come with an explicitly stated condition. Intuitively, $fA \Rightarrow O(B)$ means that A implies that B is obligatory, as long as the usual assumptions about A are true. The $\mathcal{S}$CIFF$^{\mathcal{D}}$ semantics implements an implicit assumption that an expectation is not discharged (and is therefore required to be fulfilled), which can be defeated by an explicit dischargement atom (which allows for the expectation not to be fulfilled).

Brown [47] proposes diachronic deontic logic as a language to model an agent's obligations, which have a time associated to them and can evolve over time depending also on the agent's actions. Agency is accommodated by a forward branching time model. Note that, in Brown's work, an obligation is discharged when it ceases to be true in a history, which can be caused, for example, by an action that fulfills the obligation, or by one that makes it irrelevant. In other words, our notions of fulfillment and dischargement would coincide. Diachronic deontic logic is very expressive in a single-agent scenario, but it does not support multiple agents; also, to the best of our knowledge a complete axiomatization for it has not been provided (and in the article, the author expresses his doubts about the feasibility of an axiomatization for the complete system.).

A different approach is the combination of temporal logics with deontic logics. As an example, Ågotnes et al. [48] define Normative Temporal Logic (NTL), which replaces the standard operators of the well-known CTL [49] with deontic operators. The use of time, which forces the sequentiality of the constraints, avoids many paradoxes typical of standard deontic logic, such as those involving contrary-of-duty. Moreover, the authors present the Simple Reactive Modules Language (SRML) which follows NTL and allows the execution of model checking in four different scenarios depending on the presence or absence of an interpretation of the normative system and on the definition of the model under examination. Similarly, $\mathcal{S}$CIFF$^{\mathcal{D}}$ can manage time although it does not follow the temporal logic semantics. A similar approach is proposed by the same authors in [50] where they present the Norm Compliance CTL (NCCTL). This logic extends CTL by adding a new deontic-like operator P modeling coalitions between norms which cooperate in the normative system. NCCTL is equipped with a model checker, called NORMC [51]. Since $\mathcal{S}$CIFF performs on-the-fly checking of compliance, the two systems cannot be directly compared.

Although our work does not deal with conflicting expectations and their resolution, nor with contrary-to-duty obligations, in the following we review some important contributions, which contain ideas that may inspire extensions of our framework in future work.

Many proposals consider the fact that there may be conflicting rules given a certain knowledge, bringing knowledge after their application into an inconsistent state. A possible way to solve this problem is to establish a priority order on the rules and take advantage of this order to decide which rules should be applied or not to reach a consistent state. In [52] the author considers the logic of imperatives and establishes a strict partial order on the imperatives. The idea is to build a state by verifying that the number of rules violated (conflicting rules that have not been applied) is as low as possible and that these rules are (possibly) all of lower priority than those applied. The paper presents several ways to build such states. The main objective followed during the discussion is to avoid conflicts when not necessary. Following this aim, a new operator is defined such that for each imperative, the operator adds in the set the imperative itself and all the imperatives with a higher

priority that are not violated by the current state.

A similar approach is given by [53]. The first difference is that while in [52] the state is built only by adding new facts, in [53] it is possible to remove already asserted facts if, given the current state, we find a set of rules S already applied to the state that can be replaced by a set of other rules D not yet applied which (1) all have higher priority than those in S and (2) their application to the current state (i.e., without removing consequents of the rules in S) leads to a conflict. The set S is defined as the minimal set of rules that must be withdrawn from the current state in order to safely apply rules in D.

In [54], the author extends [53] by considering the premises "it ought to be something" as having the same results in the reasoning of the premises "there are good reasons to do something" and demonstrates that the use of this extension allows the correct handling of classical reasoning problems for deontic logics such as disjunctive syllogism and disjunction introduction. While the author considers only the work of [53] in the article, his extension can also be applied directly to [52].

This paper shows that representing and reasoning with norms facilitate for the adoption of such approaches in many scenarios. Checking whether certain facts are compliant with the normative system in use is in fact often needed. Abductive frameworks such as $\mathcal{S}$CIFF$^{\mathcal{D}}$ can also be used, for example, in forensics for analysing and arguing on evidence of a crime or for explaining causal stories, sequence of states and events forming (part of) a case. Such stories must be coherent and there must be a process, usually abductive, able to prove their truthfulness. These necessities are pointed out for example by Bex et al. [55, 56], who present a hybrid framework that combines the two most used approaches in reasoning about criminal evidences: argumentative and story-based analysis. Both of them could benefit from the use of normative systems.

The rules in the $\mathcal{S}$CIFF language typically relate happened events with expectations. In the literature, rules linking directly happened events (intended as *brute facts*) with obligations / prohibitions are named *regulative norms*. Beside those there exist also *constitutive norms*, in which brute facts can produce institutional changes (such as making of contracts, or decreeing of marriages / divorces) from which in turn obligations or prohibitions can be introduced. As explained by Grossi and Jones [41], *terms such as claim, right, duty, ownership [...] allow us to connect a set of concrete circumstances to a set of legal or, more generally, normative consequences.* Avoiding such terms is possible, but the price to be paid is that many more rules could be necessary. For example (taken from [41]) one could have that n different brute facts could have the same institutional effect, and that introduces a set of m normative consequences. Linking directly brute facts with deontic behavior would need $n \cdot m$ regulative norms. Instead, one could introduce one of the previous terms and link n rules to the institutional fact (e.g., creation of a contract), that itself implies the normative (deontic) consequences, with only $n + m$ rules. After Searle [57], constitutive norms have often been presented as *count-as conditionals*, in which action X counts as Y in a context C. The $\mathcal{S}$CIFF proof-procedure, beside linking directly happened events and expected behavior, can also produce other generic abducibles, that might be used as constitutive norms.

It can be the case that two or more obligations cannot be mutually realized, giving rise to a deontic conflict. In the face of such conflicts, Standard Deontic Logic (henceforth SDL) leads to triviality in view of the rule (D): from OA , infer $\neg O \neg A$. Giving up (D) is necessary but not sufficient to allow for deontic conflicts: whenever the premises feature a deontic conflict, the other rules of SDL still cause

deontic explosion, i.e. the conclusion that everything is obligatory.

As discussed in [58], to solve this problem, various conflict-tolerant deontic logics have been developed, usually in a monotonic setting. But there is also a variety of non- monotonic formalizations that are conflict-tolerant and give rise to stronger consequence relations (e.g. Input/Output logics with constraints among them). However, these approaches typically lack a proof theory that explicates the (dynamics of) reasoning on the basis of deontic conflicts. In [58], the authors presents a logic (named $\mathbf{MP}_{\sqsubset}$) that explicates non-monotonic reasoning for handling prioritized obligations. Starting from premise sets consisting of possibly conflicting prima facie obligations that have a modular order, $\mathbf{MP}_{\sqsubset}$ allows to derive the actual, all-things-considered obligations from such premise sets. The language of $\mathbf{MP}_{\sqsubset}$ contains an infinite number of conflict-tolerant ought-operators: $O_1, O_2, O_3, \ldots$. The formula $O_i A$ is a prima facie obligation of priority level i that tells us to do or bring about A. The priority of the normative standard gets higher as the priority index gets lower. A model theoretic semantics and a proof systems based on Modus Ponens are defined for $\mathbf{MP}_{\sqsubset}$, with sound and completeness results. The authors also state other (meta) properties and show that $\mathbf{MP}_{\sqsubset}$ can be also used as basis of an adaptive logic for (prioritized) belief base revision. In our work, inconsistency among raised expectations can occur, and this leads – from an operational point of view – to failure. There is no notion of priority among expectations. Nonetheless, the withdrawing of expectations – due to some rule or possibly norm - introduces a non-monotonic flavor in our normative reasoning system, but then the underlying proof procedure – suitably extended – treats the abduced literals (i.e. expectations, their withdrawing, and their intertwining) in a monotonic manner, by always adding them to the (monotonic) set of abduced literals.

Orderings among (propositional) formulae were introduced in order to cope with a logic of preferences in deontic logic. Statements of dyadic obligation like "it ought to be the case that ϕ under condition ψ" were interpreted in terms of a binary relation $\preceq$ between states according to a certain maximality-based semantics (see [59] for a summary). Depending on the properties of the relation $\preceq$ different logics can be obtained. In [59], the authors extend maximality-based ordering models and their applications to deontic logic by recovering this semantics in the area of preference logic. They represent norms in presence of different (ordered) states by priority sequences, like the following one:

$$(\neg t \vee m \vee \neg m) \prec (\neg t \vee m) \prec \neg t$$

representing the often cited quote from St Paul's First Letter to the Corinthians: *"It is good for a man not to touch a woman. But if they cannot contain, let them marry: for it is better to marry than to burn."* (cf. [60], p. 6). Priority sentences therefore give an order to sentences, and syntactically represent (semantical) preferences among states and models. In [59] the authors not only expand semantic models, enriched with syntactic priority structure, but also equip agents with reasoning capabilities about their obligations. To this purpose, they present a simple logic capturing reasoning with betterness orderings induced by priority sentences (and their underlying data structures named priority graphs).

In our work, we mainly deal with discharging expectations and their possible reinstatement. In principle, in SCIFF one could deal with recovery from failure by disabling the SCIFF transitions dealing with violations; then the recovery from failure due to the violation of some expectation can be achieved by applying alternative constraints, fired in the new scenario, provided that inconsistent

expectations are discharged. For instance, with reference to the example above, we could write:

$true \rightarrow \mathbf{EN}(t)$

$\mathbf{H}(t) \rightarrow \mathbf{D}(\mathbf{EN}(t)), \mathbf{E}(m)$

$\mathbf{H}(t), \neg(\mathbf{H}(m)) \rightarrow \mathbf{D}(\mathbf{E}(m)), \mathbf{E}(b)$

where t, m and b stand for touch, marry and burn. Therefore, we can represent the three alternatives of the original sentence, while preserving the ordering among them.

6. Conclusions

In this article we continue our line of research that applies abductive logic programming to the formalization of normative systems.

We introduced the $\mathcal{S}$CIFF$^{\mathcal{D}}$ language, extending the $\mathcal{S}$CIFF abductive framework with the notion of dischargeable obligation. Dischargeable obligations can occur in the head of forward rules (named ICs), fired under specific conditions mentioned in the body of the rules.

Moreover, we address the intriguing case in which discharged obligations are revived, a situation that can happen in real life, e.g. when a nullified contract is later found out to be valid. In the new semantics, an expectation can be discharged and re-stated an unbound number of times, all in a declarative way.

The $\mathcal{S}$CIFF$^{\mathcal{D}}$ declarative semantics and its operational counterpart for verification accordingly extend $\mathcal{S}$CIFF's, and soundness is proved under syntactic conditions over these (discharging) constraints.

To experiment with the framework, we considered case studies requiring the notion of discharging of an obligation. In particular, we considered the articles in the Japanese Civil Code that deal with the rescission of manifestation of intentions and prevention of fulfillment of conditions. We also show - informally - the result of running the operational support upon this example in some simple scenarios.

Acknowledgements

This work is the result of a collaboration, including reciprocal visits, between the National Institute of Informatics, Tokyo, Japan, and the University of Ferrara, Italy.

This work was partially supported by the "GNCS-INdAM".

References

[1] Sergot MJ, Sadri F, Kowalski RA, Kriwaczek F, Hammond P, Cory HT. The British Nationality Act as a logic program. *Commun. ACM*, 1986. **29**:370–386. doi:http://doi.acm.org/10.1145/5689.5920.

[2] Boella G, van der Torre L, Verhagen H. Introduction to normative multiagent systems. *Comput. Math. Organ. Th.*, 2006. **12**:71–79.

[3] von Wright G. Deontic logic. *Mind*, 1951. **60**:1–15.

[4] Prakken H, Sartor G. Argument-Based Extended Logic Programming with Defeasible Priorities. *J. Appl. Non-Classical Logics*, 1997. **7**(1):25–75. doi:10.1080/11663081.1997.10510900.

[5] Governatori G, Rotolo A. BIO logical agents: Norms, beliefs, intentions in defeasible logic. *Auton. Agent Multi-Ag.*, 2008. **17**(1):36–69.

[6] Satoh K, Asai K, Kogawa T, Kubota M, Nakamura M, Nishigai Y, Shirakawa K, Takano C. PROLEG: An Implementation of the Presupposed Ultimate Fact Theory of Japanese Civil Code by PROLOG Technology. In: Onada et al. [61], 2010 pp. 153–164. doi:10.1007/978-3-642-25655-4_14.

[7] De Vos M, Padget J, Satoh K. Legal Modelling and Reasoning Using Institutions. In: Onada et al. [61], 2011 pp. 129–140. doi:10.1007/978-3-642-25655-4_12.

[8] Dignum V, Meyer JJ, Weigand H, Dignum F. An Organizational-Oriented Model for Agent Societies. In: 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002). ACM Press, 2002 .

[9] Dignum V, Meyer JJ, Weigand H. Towards an Organizational Model for Agent Societies Using Contracts. In: Castelfranchi C, Lewis Johnson W (eds.), 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part II. ACM Press, Bologna, Italy. ISBN 1-58113-480-0, 2002 pp. 694–695.

[10] ALFEBIITE: A Logical Framework for Ethical Behaviour between Infohabitants in the Information Trading Economy of the Universal Information Ecosystem. IST-1999-10298, 1999.

[11] Arisha KA, Ozcan F, Ross R, Subrahmanian VS, Eiter T, Kraus S. IMPACT: a Platform for Collaborating Agents. *IEEE Intell. Syst.*, 1999. **14**(2).

[12] Eiter T, Subrahmanian V, Pick G. Heterogeneous active agents, I: Semantics. *Artif. Intell.*, 1999. **108**(1-2):179–255.

[13] Bracciali A, Demetriou N, Endriss U, Kakas AC, Lu W, Mancarella P, Sadri F, Stathis K, Terreni G, Toni F. The KGP Model of Agency for Global Computing: Computational Model and Prototype Implementation. In: Priami C, Quaglia P (eds.), Global Computing, volume 3267 of *LNCS*. Springer. ISBN 3-540-24101-9, 2004 pp. 340–367.

[14] Alberti M, Chesani F, Gavanelli M, Lamma E, Mello P, Torroni P. Verifiable Agent Interaction in Abductive Logic Programming: the *S*CIFF Framework. *ACM T. Comput. Log.*, 2008. **9**(4):29:1–29:43.

[15] Alberti M, Gavanelli M, Lamma E, Mello P, Sartor G, Torroni P. Mapping Deontic Operators to Abductive Expectations. *Comput. Math. Organ. Th.*, 2006. **12**(2–3):205 – 225. doi:10.1007/s10588-006-9544-8.

[16] Sadri F, Stathis K, Toni F. Normative KGP agents. *Comput. Math. Organ. Th.*, 2006. **12**(2-3):101–126.

[17] Broersen J, Cranefield S, Elrakaiby Y, Gabbay D, Grossi D, Lorini E, Parent X, van der Torre LWN, Tummolini L, Turrini P, Schwarzentruber F. Normative Reasoning and Consequence. In: Andrighetto G, Governatori G, Noriega P, van der Torre LWN (eds.), Normative Multi-Agent Systems, volume 4 of *Dagstuhl Follow-Ups*, pp. 33–70. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany. ISBN 978-3-939897-51-4, 2013. doi:10.4230/DFU.Vol4.12111.33. URL `http://drops.dagstuhl.de/opus/volltexte/2013/3999`.

[18] Kakas AC, Kowalski RA, Toni F. Abductive Logic Programming. *J. Logic Comput.*, 1993. **2**(6):719–770.

[19] Gavanelli M, Lamma E, Riguzzi F, Bellodi E, Zese R, Cota G. An abductive Framework for Datalog+-Ontologies. In: De Vos M, Eiter T, Lierler Y, Toni F (eds.), Technical Communications of the 31st International Conference on Logic Programming (ICLP 2015), number 1433 in CEUR-WS. Sun SITE Central Europe, Aachen, Germany, 2015 .

[20] Gavanelli M, Lamma E, Riguzzi F, Bellodi E, Zese R, Cota G. Abductive Logic Programming for Datalog± Ontologies. In: Ancona D, Maratea M, Mascardi V (eds.), Proceedings of the 30th Italian Conference on Computational Logic (CILC2015), Genova, Italy, 1-3 July 2015, number 1459 in CEUR Workshop Proceedings. Sun SITE Central Europe, Aachen, Germany, 2015 pp. 128–143. URL http://ceur-ws.org/Vol-1459/paper21.pdf.

[21] Fung TH, Kowalski RA. The IFF proof procedure for abductive logic programming. *J. Logic Program.*, 1997. **33**(2):151–165.

[22] Jaffar J, Maher MJ. Constraint Logic Programming: A Survey. *J. Logic Program.*, 1994. **19/20**:503–581.

[23] Gavanelli M, Alberti M, Lamma E. Integration of Abductive Reasoning and Constraint Optimization in SCIFF. In: Hill PM, Warren DS (eds.), Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings, volume 5649 of *Lecture Notes in Computer Science*. Springer. ISBN 978-3-642-02845-8, 2009 pp. 387–401. doi:10.1007/978-3-642-02846-5_32. URL https://doi.org/10.1007/978-3-642-02846-5_32.

[24] Alberti M, Gavanelli M, Lamma E. Deon+ : Abduction and Constraints for Normative Reasoning. In: Artikis A, Craven R, Cicekli NK, Sadighi B, Stathis K (eds.), Logic Programs, Norms and Action - Essays in Honor of Marek J. Sergot on the Occasion of His 60th Birthday, volume 7360 of *LNCS*. Springer. ISBN 978-3-642-29413-6, 2012 pp. 308–328. doi:10.1007/978-3-642-29414-3_17.

[25] Gavanelli M, Alberti M, Lamma E. Accountable Protocols in Abductive Logic Programming. *ACM Transactions on Internet Technology*, 2018. **18**(4):46:1–46:20. doi:10.1145/3107936. URL http://doi.acm.org/10.1145/3107936.

[26] Alberti M, Chesani F, Gavanelli M, Lamma E, Mello P, Montali M. An Abductive Framework for A-Priori Verification of Web Services. In: Maher M (ed.), 8th Symposium on Principles and Practice of Declarative Programming. ACM Press, New York, USA. ISBN 1-59593-388-3, 2006 pp. 39–50.

[27] Gavanelli M, Lamma E, Riguzzi F, Bellodi E, Zese R, Cota G. Abductive Logic Programming for Normative Reasoning and Ontologies. In: Otake M, Kurahashi S, Ohta Y, Satoh K, Bekki D (eds.), New Frontiers in Artificial Intelligence (JSAI-isAI 2015 Workshops, LENLS, JURISIN, AAA, HAT-MASH, TSDAA, ASD-HR and SKL, Kanagawa, Japan, November 16-18, 2015, Revised Selected Papers), volume 10091 of *Lecture Notes in Artificial Intelligence*. Springer, 2017 pp. 187–203.

[28] Gavanelli M, Lamma E, Riguzzi F, Bellodi E, Zese R, Cota G. Reasoning on Datalog$^{\pm}$ Ontologies with Abductive Logic Programming. *Fundamenta Informaticae*, 2018. **159**(1-2):65–93. doi:10.3233/FI-2018-1658. URL https://doi.org/10.3233/FI-2018-1658.

[29] Alberti M, Gavanelli M, Lamma E, Riguzzi F, Zese R. Dischargeable Obligations in Abductive Logic Programming. In: Costantini S, Franconi E, Von Woensel W, Kontchakov R, Sadri F, Roman D (eds.), Rules and Reasoning - International Joint Conference, RuleML+RR 2017, London, UK, July 12-15, 2017, Proceedings, volume 10364 of *Lecture Notes in Computer Science*. Springer, 2017 pp. 7–21.

[30] Alchourrón CE, Gärdenfors P, Makinson D. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *J. Symbolic Logic*, 1985. **50**(2):510–530.

[31] Japanese Civil Code, part I. https://en.wikisource.org/wiki/Civil_Code_of_Japan/Part_I. Retrieved on July 19, 2016.

[32] Kunen K. Negation in logic programming. *The Journal of Logic Programming*, 1987. **4**(4):289 – 308. doi:https://doi.org/10.1016/0743-1066(87)90007-0. URL http://www.sciencedirect.com/science/article/pii/0743106687900070.

[33] Clark KL. Negation as failure. In: Logic and data bases. Springer, 1978 pp. 293–322.

[34] Eiter T, Gottlob G. The Complexity of Logic-based Abduction. *J. ACM*, 1995. **42**(1):3–42. doi:10.1145/200836.200838.

[35] Ryu YU, Lee RM. Defeasible Deontic Reasoning: A Logic Programming Model. In: Meyer JJ, Wieringa R (eds.), Deontic Logic in Computer Science: Normative System Specification, pp. 225–241. John Wiley & Sons Ltd, 1993.

[36] Kowalski RA, Sergot MJ. A Logic-based Calculus of Events. *New Generat. Comput.*, 1986. **4**(1):67–95. doi:10.1007/BF03037383.

[37] Gebser M, Kaufmann B, Kaminski R, Ostrowski M, Schaub T, Schneider MT. Potassco: The Potsdam Answer Set Solving Collection. *AI Commun.*, 2011. **24**(2):107–124. doi:10.3233/AIC-2011-0491.

[38] Kowalski R, Satoh K. Obligation as Optimal Goal Satisfaction. *Journal of Philosophical Logic*, 2018. **47**(4):579–609. doi:10.1007/s10992-017-9440-3. URL `https://doi.org/10.1007/s10992-017-9440-3`.

[39] Makinson D, van der Torre LWN. Input/Output Logics. *J. Philosophical Logic*, 2000. **29**(4):383–408. doi:10.1023/A:1004748624537. URL `https://doi.org/10.1023/A:1004748624537`.

[40] Parent X, van der Torre L. Handbook of Deontic Logic and Normative Systems, chapter Input/output Logic. In: Gabbay et al. [62], 2013.

[41] Grossi D, Jones AJ. Handbook of Deontic Logic and Normative Systems, chapter Constitutive Norms and Counts-as Conditionals. In: Gabbay et al. [62], 2013.

[42] Boella G, van der Torre LWN. Permissions and Obligations in Hierarchical Normative Systems. In: Zeleznikow J, Sartor G (eds.), 9th International Conference on Artificial Intelligence and Law, ICAIL 2003, Edinburgh, Scotland, UK, Proceedings. ACM Press, 2003 pp. 109–118.

[43] Governatori G, Rotolo A, Sartor G. Temporalised Normative Positions in Defeasible Logic. In: Proceedings of the 10th International Conference on Artificial Intelligence and Law, ICAIL '05. ACM, New York, NY, USA. ISBN 1-59593-081-7, 2005 pp. 25–34. doi:10.1145/1165485.1165490. URL `http://doi.acm.org/10.1145/1165485.1165490`.

[44] Governatori G, Milosevic Z, Sadiq SW. Compliance checking between business processes and business contracts. In: 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC), 2006, Hong Kong, China. IEEE Computer Society, 2006 pp. 221–232.

[45] Governatori G. Representing business contracts in *RuleML. Int. J. Coop. Inf. Syst.*, 2005. **14**(2-3):181–216.

[46] Alchourrón CE. Detachment and defeasibility in deontic logic. *Studia Logica*, 1996. **57**(1):5–18. doi:10.1007/BF00370667.

[47] Brown MA. Doing As We Ought: Towards A Logic of Simply Dischargeable Obligations. In: Brown MA, Carmo J (eds.), Deontic Logic, Agency and Normative Systems. Springer London, London. ISBN 978-1-4471-1488-8, 1996 pp. 47–65.

[48] Ågotnes T, van der Hoek W, Rodríguez-Aguilar JA, Sierra C, Wooldridge M. On the Logic of Normative Systems. In: Veloso MM (ed.), IJCAI 2007, volume 7. AAAI Press/IJCAI, 2007 pp. 1175–1180.

[49] Emerson EA. Temporal and Modal Logic. In: Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B), pp. 995–1072. Elsevier, 1990.

[50] Ågotnes T, van der Hoek W, Wooldridge M. Robust normative systems and a logic of norm compliance. *Log. J. IGPL*, 2010. **18**(1):4–30.

[51] Kazmierczak P, Pedersen T, Ågotnes T. NORMC: a Norm Compliance Temporal Logic Model Checker. In: Kersting K, Toussaint M (eds.), 6th Starting AI Researchers' Symposium, STAIR 2012, Montpellier, France, volume 241 of *FRONTIERS*. IOS Press, 2012 pp. 168–179.

[52] Hansen J. Prioritized conditional imperatives: problems and a new proposal. *Auton. Agent Multi-Ag.*, 2008. **17**(1):11–35. doi:10.1007/s10458-007-9016-7. URL https://doi.org/10.1007/s10458-007-9016-7.

[53] Horty JF. Defaults with Priorities. *J. Philos. Logic*, 2007. **36**(4):367–413. doi:10.1007/s10992-006-9040-0. URL https://doi.org/10.1007/s10992-006-9040-0.

[54] Nair S. Consequences of reasoning with conflicting obligations. *Mind*, 2014. **123**(491):753–790.

[55] Bex F, Prakken H, Reed C, Walton D. Towards a Formal Account of Reasoning about Evidence: Argumentation Schemes and Generalisations. *Artif. Intell. Law*, 2003. **11**(2-3):125–165.

[56] Bex FJ, van Koppen PJ, Prakken H, Verheij B. A hybrid formal theory of arguments, stories and criminal evidence. *Artif. Intell. Law*, 2010. **18**(2):123–152.

[57] Searle JR. Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, 1969. ISBN 9781139173438. doi:10.1017/CBO9781139173438. URL https://doi.org/10.1017/CBO9781139173438.

[58] Van De Putte F, Straßer C. A Logic for Prioritized Normative Reasoning. *J. Log. and Comput.*, 2013. **23**(3):563–583. doi:10.1093/logcom/exs008. URL http://dx.doi.org/10.1093/logcom/exs008.

[59] van Benthem J, Grossi D, Liu F. Priority Structures in Deontic Logic. *Theoria*, 2014. **80**(2):116–152. doi:10.1111/theo.12028. https://onlinelibrary.wiley.com/doi/pdf/10.1111/theo.12028, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/theo.12028.

[60] Fraassen BCV. Values and the Heart's Command. *Journal of Philosophy*, 1973. **70**(1):5–19. doi:10.2307/2024762.

[61] Onada T, Bekki D, McCready E (eds.). New Frontiers in Artificial Intelligence - JSAI-isAI 2010 Workshops, volume 6797 of *LNCS*. Springer, 2011. ISBN 978-3-642-25654-7. doi:10.1007/978-3-642-25655-4.

[62] Gabbay D, Horty J, Parent X, van der Meyden R, van der Torre L (eds.). Handbook of Deontic Logic and Normative Systems. College Publications, 2013.