# Designing for Compliance: Norms and Goals

Guido Governatori[2], Francesco Olivieri[2,1,3],
Simone Scannapieco[2,1,3], and Matteo Cristani[1]

[1] Department of Computer Science, University of Verona, Italy
[2] NICTA, Queensland Research Laboratory, Australia
[3] Institute for Integrated and Intelligent Systems, Griffith University, Australia

**Abstract.** We address the problem of define a modal defeasible theory able to capture intuitions as "being compliant" with a set of *norms* and a set of *goals*. We will treat norms and goals as modalised literals. From the definition of this new kind of logic, two main issues arises whether a theory is compliant or not: (a) how to revise a non compliant theory to obtain a new compliant one; (b) in case the theory is compliant how to create an entirely new process starting from the theory, i.e., from the fully declarative description of the specifications for a process and the norms.

## 1 Introduction

Business process modelling technology (BPM) emerged as a strong paradigm for the modelling, analysis, improvement, and automation of the day-to-day activities of organisations. The field is now a mature research field with a widespread adoption in industry. BPM covers a wide variety of methodologies; from graphical modelling languages to ease the understanding of the stakeholders (e.g., EPC, BPMN) to fully precise mathematical formalisms (e.g., Petri Nets, $\pi$-calculus) for formal analysis of the properties and automated verification of the processes.

Most of the existing approaches in the field are *procedural*: they point out step by step what to do in many different scenarios. If from one side this procedural nature is their strength, it is also their main drawback. In fact, they suffer two main limitations: such a paradigm is not suitable to capture flexible business processes (BPs), i.e., processes whose internal structure and relationships among the various tasks is dynamic and with a large degree of variations. Secondly, it is hard to obtain precise information about the order of the actions to be performed from the business requirements.

To obviate these problems, the trend of modelling processes by *declarative specifications* gained momentum. Instead of specifying a process step by step, the focus in this approach is on defining relationships among the tasks to be executed to achieve a goal. Examples are temporal relationships between tasks (e.g., before, after), co-occurrence/absence, dependency and so on. For a seminal work in this area see [1]. Thus, in this paradigm there is a switch from *how* (procedural) to *what* (declarative).

Another crucial aspect in the recent investigation on BPM is on regulatory compliance, again an area where the focus is on *what* a BP does. Compliance is the study of the *norm* regulating the organisational environment. Norms from a regulative source represent the perfect example of declarative specifications [2], and the related topic of

F. Olken et al. (Eds.): RuleML 2011 - America, LNCS 7018, pp. 282–297, 2011.
© Springer-Verlag Berlin Heidelberg 2011

*norm compliance* has consequently become a crucial issue in BP design and verification. Nowadays, the study of compliance focused only on norms, defining formalisms to express them, and conditions under which a system described with such formalisms can be considered *compliant* with the norms. Thus, to the best of our knowledge, there is a lack of formalisms describing systems to be compliant with a set of *goals*, even if this topic appears to be of main interest[1].

The issue is to extend the existing formalisms to the new concept of *goal compliance*: the motivation is that, given a system, it is easy to give criteria to be compliant with a set of norms but such that they do not allow to get the final goals of the enterprise.

Let us consider the following example. We have to manage the BP of a library. A norm could state that each book must be returned to the library before a fixed deadline. One of the (non-plausible) policy to be compliant with this (plausible) norm is not to lend books anymore (in clear contrast with the final aim of a library).

The aim of this work is to formally define a framework that integrates norm compliance with goal compliance, in such a way that declarative specifications of a BP could satisfy at the same time the goals of the organisation, and the norms governing the business. The contribution of the paper is twofold: First of all, we explicitly introduce the notion of goal compliance. Additionally, we work with fully declarative specifications of the capabilities of an organisation, its goals and the norms regulating the underlying business. Accordingly, we have to depart from the algorithms devised to determine whether a procedural BP is compliant, and we have to introduce the notion of violation. Given that we model the various notions in modal defeasible logic, we show how to extend the proof mechanisms of the logic to identify non-compliant situations.

The layout of the paper is as follows. In Section 2 we formalise the framework adopted, based on a particular type of Modal Defeasible Logic. Such a formalism is the tool exploited in Section 3 to integrate the concepts of norm and goal in a business process. The main core of this work is Section 4 where we present theoretical properties and definitions of *norm and goal compliance*. In there, we also outline possible future research developments.

## 2 Logics

In this section we are going to introduce the logic we use to model processes, their organisational goals, and the norms governing them. The logic is an extension of Defeasible Logic (DL) [3]. In particular, it extends and combines the deontic DL of violations [4] for modelling contracts and then used for regulatory compliance of processes [5,6], and the defeasible BIO (Belief-Intention-Obligation) logic for modelling agents [7].

### 2.1 Language of Modal Defeasible Logic

The main aim of this subsection is to build an inference process to compute factual knowledge (through belief rules), goals and obligations from existing facts, primitive

[1] In here we follow the classical definitions of norm and goal as give in the literature for agents where a norm is an external constraint while a goal is a internal one.

goals and unconditional obligations. As a first step, we build a defeasible theory whose basic elements are (1) a set of *facts* or indisputable statements, (2) three sets of rules: for beliefs, goals, and obligations, (3) a superiority relation to determine the relative strength of conflicting rules. Thus, a defeasible theory $T$ is a structure $(F, R, >)$, where $F$ is the set of facts, $R$ is the set of rules, and $>$ is a binary relation over $R$.

*Belief rules* are used to relate the factual knowledge of an enterprise, composed by

- the set of *actions* (or *tasks*) an organisation can do;
- the *pre-conditions* under which tasks can be executed;
- the *effects* derived by the execution of these tasks (also called *post-conditions*).

Specifically, belief rules describe the logical relationship between pre-conditions and tasks, tasks and their effects, relationships between tasks, relationships between states. As such, provability for beliefs does not generate modalised literals. *Obligation rules* determine when and which obligations are in force. The conclusions generated by obligation rules are modalised with obligation. Finally, *goal rules* establish the goals of an organisation depending on the particular context. Accordingly, similar to obligation rules, the conclusions of this type of rules take the goal modality.

Following ideas given in [8], rules can gain more expressiveness when a *preference operator* $\odot$ is used, whose meaning will be clearer in the remainder. Intuitively, an expression like $A \odot B$ means that if $A$ is possible, then $A$ is the first choice, and $B$ is the second one; if $\neg A$ holds, then the first choice is not attainable and $B$ is the actual choice.

We now introduce the language adopted in the rest of the paper to make precise the above mentioned ideas. Let PROP be a set of propositional atoms, $\mathrm{MOD} = \{\mathrm{B}, \mathrm{G}, \mathrm{O}\}$ the set of modal operators, Lab be a set of arbitrary labels, and $\mathrm{Act} = \{t_1, \dots, t_n\} \subseteq$ PROP be a set of basic actions (or tasks). The set $\mathrm{Lit} = \mathrm{PROP} \cup \{\neg p | p \in \mathrm{PROP}\}$ denotes the set of *literals*. The *complementary* of a literal $q$ is denoted by $\sim q$; if $q$ is a positive literal $p$, then $\sim q$ is $\neg p$, and if $q$ is a negative literal $\neg p$ then $\sim q$ is $p$. The set of modal literals is $\mathrm{ModLit} = \{Xl, \neg Xl | l \in \mathrm{Lit}, X \in \{\mathrm{G}, \mathrm{O}\}\}$[2]. We introduce two preference operators, $\otimes$ for obligations and $\oplus$ for goals, and we will use $\odot$ when we refer to one of them generically. These operators are used to build chains of preferences, called $\odot$-expressions. The formation rules for $\odot$-expressions are: (a) every literal is an $\odot$-expression, (b) if $A$ is an $\odot$-expression and $b$ is a literal then $A \odot b$ is an $\odot$-expression. In addition we stipulate that $\otimes$ and $\oplus$ obey the following properties: (1) $a \odot (b \odot c) = (a \odot b) \odot c$ (associativity); (2) $\bigodot_{i=1}^{n} a_i = (\bigodot_{i=1}^{k-1} a_i) \odot (\bigodot_{i=k+1}^{n} a_i)$ where exists $j$ such that $a_j = a_k$ and $j < k$ (duplication and contraction on the right). Such $\odot$-expressions can be given both by the process designer, and can be obtained through *construction rules* based on the particular logic adopted REF.

We adopt the classical definitions of *strict rules*, *defeasible rules*, and *defeaters* in DL [9]. However, for the sake of simplicity, and to better focus on the non-monotonic aspects that DL offers, in the remainder we use only defeasible rules. In addition we have to take the modal operators into account. A defeasible rule is an expression $r : a_1, \dots, a_n \Rightarrow_X c$, where

[2] For the belief modal operator B, we assume that the "$X$" modal operator is the empty modal operator, thus essentially a modal literal B$l$ is equivalent to the unmodalised literal $l$.

1. $r \in \mathrm{Lab}$ is the name of the rule;
2. $a_1, \dots, a_n$, the *antecedent* of the rule, is the set of the premises of the rule (alternatively, it can be understood as the conjunction of all the literals in it). Each $a_i$ is either a literal or a modal literal;
3. $X \in \mathrm{MOD}$ represents the type of modality introduced by the rule itself (from now on, we omit the subscript B in rules for beliefs, i.e., $a_1, \dots, a_n \Rightarrow c$ will be used as a shortcut for $a_1, \dots, a_n \Rightarrow_{\mathrm{B}} c$);
4. $C$ is the *consequent* (or *head*) of the rule, which is an $\otimes$-expression involving literals in rules for obligations, an $\oplus$-expression involving literals in rules for goals, and a single literal in rules for beliefs[3].

Several obvious abbreviations on sets of rules can be used. For example, $R^X$ ($R^X[q]$) denotes all rules introducing modality $X$ (with consequent $q$), and $R[q] = \bigcup_{X \in \mathrm{MOD}} R^X[q]$. With $R[c_i = q]$ we denote the set of rules whose head is $\otimes_{j=1}^n c_j$ for obligation rules and $\oplus_{j=1}^n c_j$ for goal rules where for some $i$, $1 \leq i \leq n$, $c_i = q$.

The meaning of $\odot$-expressions as consequent of rules is the following:

**For obligations,** a rule $a_1, \dots, a_n \Rightarrow_{\mathrm{O}} o_1 \otimes o_2 \otimes \dots \otimes o_l$ means that if conditions $a_1, \dots, a_n$ hold, then the obligation in force is $o_1$, but if $\neg o_1$ is the case, then the new obligation in force is $o_2$, and so on. Then, obeying obligation $o_l$ represents the last chance to obtain a still acceptable situation with respect to the regulative system in force, but it is not possible to recover from its violation. In deontic logic, this type of expressions, namely the activation of certain obligations in case of other obligations being violated, is referred to as *contrary-to-duty* (abbreviated CTD) obligations, or *reparation obligations* [2].

**For goals,** a rule $a_1, \dots, a_n \Rightarrow_{\mathrm{G}} g_1 \oplus g_2 \oplus \dots \oplus g_m$ means that if conditions $a_1, \dots, a_n$ hold, then we have to reach the goal $g_1$, but if it is not possible, then the goal to achieve is $g_2$, and so on. A chain for goals can be seen as a mean to express a preferential list of organisational objectives such that every goal is more restrictive than all other successive goals in the chain. As such, $g_m$ is the last acceptable outcome we expect to obtain from the business process with respect to this particular chain.

The terminology defined so far is mostly taken from [7], where an extension of DL with modal operators is introduced to differentiate modal and factual rules. However, labelling the rules of DL produces nothing more but a simple treatment of the modalities, thus two interaction strategies between modal operators are analysed:

1. *rule conversions:* using rules for a modality $X$ as they were for another modality $Y$, i.e., the possibility to convert one type of conclusions into a different one. For example, if 'a car industry has the purpose of assembling perfectly working cars' and 'it is known that in every working car there is a working engine', then 'a

[3] It is worth noting that modalised literals can occur only in the antecedent of rules: the reason is that the rules are used to derive modalised conclusions and we do not conceptually need to iterate modalities. The motivation of a single literal as a consequent for belief rules is dictated by the intended reading of the belief rules, where these rules are used to describe the environment.

car industry has also the purpose of assembling working engines in every car produced'. Formally, we define a binary relation Convert $\subseteq$ MOD $\times$ MOD such that Convert$(X,Y)$ means 'a rule of type $X$ can be used also to produce conclusions of type $Y$'. This intuitively corresponds to the following logical schema:

$$\frac{Ya_1,\dots,Ya_n \quad a_1,\dots,a_n \Rightarrow_X b}{Yb}\ \text{Convert}(X,Y)$$

2. *conflict-detection* and *conflict-resolution:* it is crucial to identify criteria for detecting and solving conflicts between different modalities. For example, if 'a tyre industry wants to produce cheaper tyres with a pollutant emission greater than *pe*', and 'by law the pollutant emission must be lower than threshold *pe*', then the tyre industry can not economise the production if she wants to obey the law. Formally, we define an asymmetric binary relation Conflict $\subseteq$ MOD $\times$ MOD such that Conflict$(X,Y)$ means 'rule types $X$ and $Y$ are in conflict and modality $X$ prevails over $Y$'. Consider the following theory:

$$a \Rightarrow_X c$$
$$b \Rightarrow_Y \neg c$$

If both $a$ and $b$ are derivable and Conflict$(X,Y)$ holds, then we derive $Xc$.

As beliefs represent the factual knowledge of the organisation, belief rules can be used to derive both obligations and goals. Thus, the following formulation of Convert arises:

$$\text{Convert} = \{(\text{B},\text{G}),(\text{B},\text{O})\}$$

Furthermore, as our main purpose is to build a new process which is compliant with a set of given norms, it seems reasonable that rules for obligations take precedence over rules for goals (i.e., an organisation tries to achieve its purposes not violating the norms). Hence, the following definition of Conflict (reflecting the behaviour of a *social agent* as described in [7]) will be used in our analysis:

$$\text{Conflict} = \{(\text{B},\text{O}),(\text{O},\text{G}),(\text{B},\text{G})\}$$

The construction of the *superiority relation* combines two components: the first $>^{sm}$ considers pairs of rules of the same modality. This component is usually given by the designer of the theory and capture the meaning of the single rules, and thus encodes the domain knowledge of the designer of the theory. The second component, $>^{\text{Conflict}}$, is obtained from the rules in a theory and depends on the meaning of the modalities. Formally, the superiority relation $>$ is such that $>=>^{sm} \cup >^{\text{Conflict}}$, where

- $>^{sm} \subseteq R^X \times R^X$ such that if $r > s$, then if $r \in \text{R}^X[p]$ then $s \in \text{R}^X[{\sim}p]$ and $>$ is acyclic;
- $>^{\text{Conflict}}$ is such that $\forall r \in \text{R}^X[p], \forall s \in \text{R}^Y[{\sim}p]$, if Conflict$(X,Y)$, then $r >^{\text{Conflict}} s$.

### 2.2 Inference in Modal Defeasible Logic

Proofs in a modal defeasible theory $T$ are linear derivations, i.e., sequences of *tagged literals* of the form $+\partial_X q$ and $-\partial_X q$. Given $X \in$ MOD, $+\partial_X q$ means that $q$ is defeasibly provable in $T$ with modality $X$, and $-\partial_X q$ means that $q$ is defeasibly refuted with modality $X$. Similarly, $\pm\partial q$ will be used as a shortcut of $\pm\partial_{\mathrm{B}} q$. The initial part of length $i$ of a proof $P$ is denoted by $P(1..i)$.

We now define when a rule is applicable or discarded. A rule for a belief is applicable if all the literals in the antecedent of the rule are provable with the appropriate modalities, while the rule is discarded if at least one of the literals in the antecedent is not provable. For the other types of rules we have to take the relation Convert into account.

**Definition 1.** *Let* Convert *be the conversion relation between elements in* MOD.

- *A rule $r$ in $R^{\mathrm{B}}$ is* applicable *iff* $\forall a \in A(r)$, $+\partial a \in P(1..n)$ *and* $\forall Xa \in A(r)$, *where* $X \in \mathrm{MOD}$, $+\partial_X a \in P(1..n)$.
- *A rule* $r \in R[c_i = q]$ *is* applicable *in the condition for* $\pm\partial_X$ *iff*
  1. $r \in R^X$, $\forall a \in A(r)$, $+\partial a \in P(1..n)$ *and* $\forall Ya \in A(r)$ $+\partial_Y a \in P(1..n)$, *or*
  2. $r \in R^Y$, $\mathrm{Convert}(Y,X)$, $A(r) \neq \emptyset$, *and* $\forall a \in A(r)$, $+\partial_X a \in P(1..n)$.
- *A rule $r$ is* discarded *if we prove either* $-\partial a$ *or* $-\partial_X a$ *for some* $a \in A(r)$.

The proof conditions for $\pm\partial_X$ are thus as follows:

$+\partial_X$: If $P(n+1) = +\partial_X q$ then
(1) $\exists r \in R[c_i = q]$ such that $r$ is applicable, and $\forall i' < i$, $-\partial c_{i'} \in P(1..n)$; and
(2) $\forall s \in R[c_j = \sim q]$, either $s$ is discarded, or $\exists j' < j$ such that $+\partial_X c_{j'} \in P(1..n)$, or
  (2.1) $\exists t \in R[c_k = q]$ such that $t$ is applicable and
    $\forall k' < k$, $-\partial c_{k'} \in P(1..n)$ and $t > s$

$-\partial_X$: If $P(n+1) = -\partial_X q$ then
(1) $\forall r \in R[c_i = q]$, either $r$ is discarded or $\exists i' < i$ such that $+\partial c_{i'} \in P(1..n)$, or
(2) $\exists s \in R[c_j = \sim q]$, such that $s$ is applicable and $\forall j' < j$, $-\partial_X c_{j'} \in P(1..n)$ and
  (2.1) $\forall t \in R[c_k = q]$ either $t$ is discarded, or
    $\exists k' < k$ such that $+\partial c_{k'} \in P(1..n)$ or $t \not> s$

Notice that the condition of provability of a literal $q$ in a chain for obligations (or goals) implies that all antecedents of the rule are provable in the theory, and that we have proved that every element $p$ prior to $q$ in the chain is not defeasibly proved (i.e., we have $-\partial p$). This means that the theory (process) fails to fulfil an obligation or achieving the goal corresponding to $p$. Given an obligation or goal rule

$$a_1, \dots, a_m \Rightarrow_X c_1 \odot \cdots \odot c_n$$

if the rule is applicable, then the obligation (and possible ways to recover from its violation) is in force (for an obligation rule) and that an organisation commits to a series of progressively less stringent alternative goals (for a goal rule). In both interpretations $c_n$ is the last chance to be compliant with the rule. In case of obligation, $c_n$ is the last thing we can obey to to result in a situation that is still deemed as legal (though not an ideal situation); in case of a goal, $c_m$ is the least of the acceptable outcomes of a process.

We are going to formally capture this intuition. To this end we introduce the literal $\perp$ whose interpretation is a non-compliant situation, and at the same time we provide proof conditions to (defeasibly) derive it. Thus, to derive a non-compliant situation ($+\partial\perp$) there is an applicable rule such that all the elements of the head are violated (for obligations) or not achieved (for goals). Conversely, to be compliant ($-\partial\perp$) for every applicable rule, at least one obligation has been complied with, or goal achieved.

More formally, for $X \in \{\mathrm{O},\mathrm{G}\}$:

$+\partial_X\bot$: If $P(n+1) = +\partial_X\bot$ then
(1) $\exists r \in R$ such that $r$ is applicable, and
(2) $\forall c_i \in C(r)\ -\partial c_i \in P(1..n)$.

$-\partial_X\bot$: If $P(n+1) = -\partial_X\bot$ then
(1) $\forall r \in R$ either $r$ is discarded or
(2) $\exists c_i \in C(r)$ such that if $-\partial c_j \in P(1..n)$ for $\forall j \leq i$, then $+\partial c_i \in P(1..n)$.

## 3 Norm and Goal Compliance

Informally, a *business process* is a collection of related activities (or *tasks*) to be performed to achieve one or more organisational *goals*. Many years of thorough analysis in the field of BPM led to the definition of several techniques apt to model and reason about BPs, from the early stages like graphical modelling (e.g., BPMN) up to the definition of running process instances starting from graphical models (e.g., BPEL).

In fact, the classical definition of BP as reported above does not take into account many factors that deeply affect the phase of process definition and maintenance. For example, when an organisation is seen as an entity embedded in an environment regulated by norms, the concept of *compliance* comes into play. In this scenario, an organisation has to take care both of the achievement of the goals it aims at (*goal compliance*), and the norms in force in the environment (*norm compliance*).

The research field resulting from the interaction of BPM approach with legal reasoning, i.e. *business process compliance*, addresses all problems regarding the alignment of the formal specifications of a (set of) BP(s), and the formal specifications of a set of norms governing the surrounding environment. Most of the research in this field regards the analysis of conditions and methodologies to check if a given (set of) BP(s) is compliant with a set of given norms. An extensive survey about the topic is given in [2], where the authors define a framework to suitably represent (1) BPs (through process graphs, e.g., in BPMN), (2) norms (using Formal Contract Logic, shortly FCL, a combination of classical DL [9] and a deontic logic of violations), and (3) the concept of norm compliance. Without entering too much into details, BP models are extended with *annotated tasks*, where annotations specifies: (1) the artefacts or *effects* of executing a task, and (2) the rules describing the norms relevant for the process.

The main result given in this work is the definition of an algorithm that:

1. traverses the graph describing the BP and identifies the sets of effects (sets of literals) for all the tasks (nodes) and propagates the effects in the process according to the execution semantics specified (token-passing mechanism as in Petri Nets [10]).
2. for each task, uses the set of effects for that particular task to determine the normative positions triggered by the execution of the task. Thus, effects of the task are used as a set of facts, and the conclusions of the defeasible theory resulting from the effects and the FCL rules annotating the task are computed. In the same way effects are accumulated, the algorithm accumulates (undischarged) obligations from one task in the process to the task following it in the process.

3. for each task, compares the effects of the tasks and the obligations accumulated up to the task. If an obligation is fulfilled by a task, the algorithm discharges the obligation, otherwise if the obligation is violated, a violation is signalled. Finally, if an obligation is not fulfilled nor violated, the obligation is kept in the stack of obligations and propagated to the successive tasks.

In this framework, the concept of goal compliance is not mentioned, but it can be trivially treated: BPs are given as an input of the compliance checking phase; as such, it seems reasonable to assume that they already achieve the goals they were built for.

Based on the same ideas about conditions a system must satisfy to be compliant with a set of norms, we are now ready to give an informal definition about what we intend for a process to be *goal compliant*. Roughly speaking, given conditions describing states of the environment we act in, we are able to perform some actions which lead us in a state of the world where all the goals we aim for are achieved. For example, if we want to go to the airport and not to be hungry, the actions of catching the train from our place to the airport, and stopping in a fast food to buy an hamburger, result in a state of the world where all our desires are fulfilled.

According to the motivations proposed in Subsection 2.1, we take in consideration norms compensating other norms, we use the same intuition also for goals: we will consider goals 'compensating' other goals, i.e., a system which fails to achieve a primary goal $A$, but succeeds in reaching goal $B$ such that $B$ is less preferred than $A$ but still acceptable, is a system that reaches a desirable state of the world.

In other terms, where contrary-to-duty chains were meant to be norms repairing the failure of the system to be adherent to previous norms, 'contrary-to-duty chains' for goals can be seen as *preference chains* [11,12]. Let us assume to have the CTD of goals $\Rightarrow_G A \oplus B$. Its meaning is that we would prefer $A$ but, if $A$ is not the case, then our second preference is for $B$ and, if also $B$ is not the case, then we have no further preferences ('we give up'). De facto, such chains form a preference partial order among elements of our theory[4]. In a situation like the previous one, while $A$ represents the most preferable goal, $B$ is not just the less preferable goal but it embodies the last chance for the system to be compliant with such a chain. $B$ can be seen as an element of the set containing all the minimal goals (the last elements in every chain for goals) to be achieved.

More formally, given a theory $T$ describing our world, with *Goals* representing the set of objectives, and *Norms* representing the set of obligations in force, a system is *norm* and *goal compliant* if

1. *norm compliant*: each norm is not violated, or if so, it is compensated (i.e., there no exists derivation in $T$ leading to an opposite of an element in *Norms* that has no further compensation);
2. *goal compliant*: the process ends having reached at least all the minimal objectives. In other words, a business process is goal compliant if there exists at least one possible way to execute it, and the execution satisfies at least one of the goal in each active goal chains.

[4] For example, we can express the fact that we have no preference between two elements, $A$ and $B$, by both $A \oplus B$ and $B \oplus A$.

We now propose an example to better explain the new concepts introduced in this section; in Section 4 where we will formalise it in our logic (Example 3).

*Example 1. PeoplEyes* is an eyeglasses manufacturer. Naturally, its final goal is to produce cool and perfectly assembled eyeglasses. The final steps of the production process are to shape the lenses to glasses, and mount them on the frames. To shape the lenses, *PeoplEyes* uses a very innovative and expensive laser machinery, while for the final mounting phase two different machineries can be used. Although both machineries work well, the first and newer one is more precise and faster than the other, so *PeoplEyes* prefers the usage of the first machinery as much as possible. Unfortunately, a new norm comes in force stating that no laser technology can be used, unless human staff wears laser-protective goggles.

If *PeoplEyes* has both human resources and raw material, and the three machineries are fully working, but it has not yet bought any laser-protective goggles, all its goals would be achieved but it would fail to comply with the regulatory, since the norm for the no-usage of laser technology is violated and not compensated.

If *PeoplEyes* buys the laser-protective goggles, its entire production process also becomes norm compliant. If, at some time, the more precise mounting machinery breaks, but the second one is still working, *PeoplEyes* still remains goal compliant since also the usage of the second machinery leads to a state of the world where the goal of mounting the glasses on the frames is reached. In this last scenario, *PeoplEyes* reaches the 'minimal set' of goals. Again, if *PeoplEyes* has no protective laser goggles and both the mounting machineries are out of order, *PeoplEyes* production process is neither norm nor goal compliant.

## 4 Designing for Compliance

We formalised in Section 2 the logic to use, while in Section 3 we informally described what we intend for a process to be *norm compliant* and *goal compliant*.

Briefly recalling those ideas, a process adheres to the sets of goals and norms when, during its execution, there is no violation of the ruling norms (or when a violation occurs, the process compensates that), and once the process ends up, all the goals are achieved (or at least a minimal set of them). Thus, to describe our processes we need

- A library of tasks: $t_1,\dots,t_n$. Tasks correspond to actions that can be performed by the system.
- Belief rules for the activation of a task $t$, and to obtain a specific condition $c$:
  - $c_1,\dots,c_m,t_1,\dots,t_n \Rightarrow t$;
  - $c_1,\dots,c_m,t_1,\dots,t_n \Rightarrow c$.[5]

[5] Here we assume that a pre-condition for a task to begin its execution can include both logical expressions and the previous execution of other tasks (*task dependency*), for example "We buy an ice cream if the sun is shining and we go to downtown". Notice that the study of dependencies among tasks is out of the scope of this work; anyway, we refer to [13] for an initial study of how to capture control flow patterns in Modal DL.

- Rules for relationships between tasks: $t_i \Rightarrow_{\mathrm{O}} t_1, \dots, t_i \Rightarrow_{\mathrm{O}} t_n$, meaning that if task $t_i$ is performed, then tasks from $t_1$ to $t_n$ must be executed as well.
- Rules describing obligations: $c_1, \dots, c_m \Rightarrow_{\mathrm{O}} \bigotimes_{i=1}^{n} a_i$, where the consequent is a reparative chain of obligations.
- Rules describing goals: $c_1, \dots, c_m \Rightarrow_{\mathrm{G}} \bigoplus_{i=1}^{n} a_i$, where the consequent is a preferential chain of goals.

In the following, when possible, we will use the subscripted literal $t$ to denote a task, and the subscripted literal $c$ to denote a condition. Given a modal defeasible theory $T = (F, R, >)$, the set of facts $F$ contains:

- literals for conditions that are to be considered true at the beginning of an instance of a BP;
- literals for tasks that can be performed with no other conditions but the ones in $F$ (or without any condition at all);

We impose that no modalised literals for norms or goals are facts, since we do not want in this preliminary analysis to consider *primitive intentions* and *unconditional obligations*. Thus, primary norms and goals are translated in our framework as the corresponding modalised rules without antecedents.

Let us consider the following example to show how a theory can fail to be compliant with respect to a modalised operator.

*Example 2.* Let $T$ be the theory with only the following rule:

$$r_1 : \Rightarrow_X a \odot b$$

This modal rule is obviously applicable, as its antecedent is empty. Furthermore, no rule whose consequent is $\neg a$ is in the theory. By definition, we obtain $+\partial_X a$. Since there are no belief rules for $a$ (nor for $b$), the theory derives, in cascade: $-\partial a$, $+\partial_X b$, $-\partial b$ and $+\partial_X \bot$. Hence, $T$ is not compliant with respect to modality $X$. If we add $b$ as a fact, or we introduce a rule like $\Rightarrow b$, then $T$ becomes compliant since we can derive $+\partial b$ (as a compensation of $a$) which leads us to derive $-\partial_X \bot$.

The ideas enlightened by the previous example underpin the conceptual bases of this work: the obligation to do a thing does not imply at all that such a thing should eventually be executed by the system. The same reasoning is valid also for goals: aiming at a goal does not result in possessing the means to reach it. Moreover, if such an obligation (goal) has no further compensation, then we definitely obtain a derivation for $\bot$ with respect to this particular chain.

The above reasoning allows us to give a formal definition of a theory to be norm compliant and goal compliant based only on the modalised derivations of $\bot$.

**Definition 2 (Norm and Goal Compliance).** *Let $T$ be a modal defeasible theory. $T$ is* norm compliant *if $T \vdash -\partial_{\mathrm{O}} \bot$ and it is* goal compliant *if $T \vdash -\partial_{\mathrm{G}} \bot$.*

*Example 3.* We now extend Example 1 formalising it in our logic.

$$
\begin{array}{ll}
F = \{\mathit{Lenses},\ \mathit{Frames}\} & \\
R = \{r_0 :\Rightarrow_{\mathrm{G}} \mathit{EyeGlasses} & r_5 : \mathit{MountingMach1} \Rightarrow \neg \mathit{MountingMach2} \\
\quad r_1 :\Rightarrow \mathit{Laser} & r_6 : \mathit{Glasses}, \mathit{MountingMach1} \Rightarrow \mathit{EyeGlasses} \\
\quad r_2 : \mathit{Lenses}, \mathit{Laser} \Rightarrow \mathit{Glasses} & r_7 : \mathit{Glasses}, \mathit{MountingMach2} \Rightarrow \mathit{Eyeglasses} \\
\quad r_3 :\Rightarrow \mathit{MountingMach1} & r_8 :\Rightarrow_{\mathrm{O}} \neg \mathit{Laser} \otimes \mathit{WearGoggles} \\
\quad r_4 :\Rightarrow \mathit{MountingMach2} & r_9 :\Rightarrow_{\mathrm{G}} \mathit{MountingMach1} \oplus \mathit{MountingMach2}\} \\
>^{sm} = \{r_4 > r_3\} &
\end{array}
$$

Since there no exists rule for goggles, the theory is goal compliant, but not norm compliant. If we add

$$r_{10} :\Rightarrow \mathit{WearGoggles}$$

to $R$ we are both norm and goal compliant, and also if we add

$$r_{11} : \mathit{OutOfOrderMountingMach1} \Rightarrow \neg \mathit{MountingMach1}$$

and *OutOfOrderMountingMach1* as a fact.

### 4.1 Revision in Case of Non Compliance

Norm and goal compliance give rise to non-trivial questions: what should we do when a BP is not norm compliant or goal compliant, or even both? Are there (efficient) ways to make a BP norm compliant once a violation of a norm occurs without affecting goal compliance? And from the other point of view, how to make a BP goal compliant once some of its goals are not achieved without affecting norm compliance? Answering these questions attains the area of *business process revision*, which has received great attention in recent years given its crucial influence on organisation practices.

Roughly speaking, all the efforts spent in this research area subscribe to two general approaches. *The first approach* relies on modelling notations defining the structural aspects of BPs, which are extended with other formalisms apt to represent the behavioural aspects. As an example, BPMN enriched with semantic annotations is able to describe the effects implied by the execution of a particular task [2]. On the same grounds, several translations from modelling notations into other formalisms have been proposed, e.g., semantic nets [14] and BP graphs [15]. *The second approach* instead is completely based on pure logical formalisms, where revising a BP means revising the logical theory describing the BP itself. The underlying theory formally represents at the same time the structure and the behaviour of the BP [13].

Describing pros and cons of both approaches is out of the scope of this paper and will be matter of future work. However, it is worth taking both into account, as they capture different (and interesting) aspects of revision. The first aims at revising a BP at an higher level, in terms of removal, addition, swapping and substitutions of tasks in the BP. On the other hand, the second one abstracts from the concepts of task and conditions that trigger (or are caused by) a task: they are all denoted by literals in the same theory, and the main focus is on how they work together to derive other literals. We suggest below one representative example of each approach, and we briefly report some hints on how the proposed ideas could be exploited for our purposes.

*Reusable modules.* The first idea, given in [16] and developed in [17], relies on the ever more emerging trend of designing BPs as related collections of *reusable* modules, i.e., set of standardised actions to be performed to achieve the goals modules have been built for (thus giving goal compliance), and that can be used with slight or no modifications also in other BPs. Modules are further augmented with built-in statements assuring that the usage of a particular module in a BP implies the norm compliance with respect to the statements specified in the module. This approach can be theoretically applicable both when a norm uncompliant process is given, or must be built from scratch and we have to assure its norm and goal compliance at design time. For the first case, we recall that the algorithm given in [2] allows to know the *exact* point in a BP where a violation of an obligation occurs. Thus, we can substitute the uncompliant part of the BP with a module that reaches the same goals and compensates the previous violation(s). In the second case, we try to build the process starting from a given repository of modules, based on goals we want to achieve and norms we have to comply with.

*Theory change via proof tags analysis.* We have already afforded the problem of revising defeasible theories by only changing the superiority relation between rules [18]. The major result is the identification of three relevant cases, named *canonical*, where a revision operator could apply only changing the relative strength between pairs of rules. More specifically, the revision operator could act on a defeasibly proved literal $p$ and makes it not provable anymore, i.e., from $+\partial p$ to $-\partial p$ (*first case*); or could act on a defeasibly proved literal $p$ and makes its opposite defeasibly proved, i.e., from $+\partial p$ to $+\partial{\sim}p$ (*second case*); or more, could act on a not defeasibly proved literal $p$ and makes it defeasibly proved, i.e., from $-\partial p$ to $+\partial p$ (*third case*). Additional proof tags other than strict ($\pm\Delta$) and defeasible proof ($\pm\partial$) for a literal are used to better study the cases. Some preliminary work suggests that proof tags analysis could represent a valid mean to categorise all possible situation an hypothetical revision operator has to cope with. Indeed, by definition of $+\partial_X\bot$, there exists at least one rule $r$ for goals or obligations such that (1) $r$ is applicable, and (2) each element $p$ in the chain is not defeasibly provable as a belief, i.e., $-\partial p$ holds. Thus, it seems reasonable that there are two ways to regain compliance with respect to the corresponding modality: we can focus on the consequent of $r$, making at least one element $p$ on the chain defeasibly provable (and falling in the third canonical case, that is from $-\partial p$ to $+\partial p$); or we can focus on the antecedent of $r$, making the rule discarded in the sense of Definition 1 (that is, making at least an antecedent $p$ not defeasibly provable with respect to its modality $X$). This is represented by a modalised variant of the first canonical case, i.e., from $+\partial_X p$ to $-\partial_X p$.

## 4.2 Compliance by Design

The idea of *Compliance by Design* is to create an entirely new process starting from a fully declarative description of the specifications for a process and the norms. These specifications are encoded in a modal defeasible theory. This theory describes the capabilities, resources of a company, and the environment the enterprise acts in; it also contains the norms governing the business and the goals the enterprise wants to achieve. We also assume such a theory to be norm and goal compliant. Thus, after the building process ends up, there is no need of checking compliance again, since the process will be compliant with norms and goals by design. Thus, from the fact that the theory we work with has been proved to be compliant, it follows that a compliant process exists. The issue is now to study methodologies to extract the graph of the process from the initial theory. Our intuition is that a derivation of a task literal corresponds to a plan leading to the achievement of the task. In this perspective the problem reduces to how to assemble together from the many derivations of the goals the corresponding plans to obtain a single business process graph.

In the current literature, two approaches appear promising for realising compliance by design. The first approach, based on the *process mining* [19,20], consists in applying the same techniques devised to induce a process graph starting from *workflow logs*, to the many derivations from the theory leading to goals and norms. In the second method, we start from the set of goals we want to achieve, and we iteratively construct the graph, rule by rule, following the structure of the theory and using its extension.

*Process Mining Approach.* [19,20] define techniques and algorithms for discovering workflow models starting from "workflow logs". A workflow log contains informations about the workflow as it is actually being executed: all the traces in a workflow log are representative and a sufficiently large subset of the possible behaviours of systems modeled in the workflows themselves. Through process mining, authors start from linear sequences of tasks to obtain complex structures capturing parallelism and choices.

We can apply the same methodologies in our case. The statement is motivated by the following reasoning. Given a reachable goal, a derivation for it is a linear sequence of (proved) literals in the theory. Thus, such a derivation can be understood as a log trace. Even if, to obtain a literal, the derivation rule has a set of premises that contains more than a single element, there exist procedures to automatically obtain derivations. For example, given the theory with the only rule $r : p, q \Rightarrow t$, where $p$ and $q$ are facts, we obtain as derivations $p \rightarrow q \rightarrow t$ and $q \rightarrow p \rightarrow t$. Being the theory compliant, the issue is to extract all the traces from such derivations, and combine them together to get a single business process graph.

*Backward Graph Approach.* BPs consist of separate activities. An activity is an action that is a semantical unit at some level. In addition, an activity can be thought of as a function that modifies the state of the process, making true some conditions, false some others, and letting some tasks to start their execution. BPs are modelled as graphs with individual activities as nodes. The edges on the graph represent the potential flow of control from one activity to another.

The modal defeasible theory we start with is rich of informations: there are literals describing tasks and conditions, rules describing the activations of tasks and their effects, reparation chains both for norms and goals. Moreover, the superiority relation states conditions under which a rule is activated (preferred) instead of another and, finally, patterns on the rules allow to identify parallel and (exclusive) choice structures.

We want to exploit all these informations to build the BP. The idea is to start from the extension of the theory (i.e., the literals that have been defeasible proved) and from the set of reachable goals, and to create a node for each goal that is a task. For each of them, we find out every rule proving it whose antecedents are all in the extension of the theory, and we store them. For every such antecedent that is also a task literal, we create a new node (if it does not exist yet) and we link it with a directed edge from it to the corresponding goal node. For every new node, we iterate the process. The procedure terminates when we reach literals for facts.

Notice that in the process described above, we never add nodes for conditions. This follows from the fact that all the conditions needed for being norm and goal compliant are already satisfied (since the initial theory is compliant). Thus, there is no need to consider literals for conditions: we must only establish which antecedents generate such literals and propagate these informations.

Since, nowadays, there is yet no algorithm computing this procedure, an empirical proof of the termination does not exist. Anyway, the compliance of the theory implies that every goal and norm is derived or compensated, and so the derivation process ends in a finite number of steps. Since the above procedure represents a "backward mirroring" of the derivation process, it also must come to an end.

## 5 Conclusions

The first contribution of this paper is the introduction of the notion of "goal compliance" and we have argued that to check whether a BP achieves the goals of an organisation can be dealt with the same methodology as "norm compliance". This provides a further motivation to subscribe to the declarative way to specify processes.

While the idea behind this work looks very natural and at the core of BPs and glimpses of it can be found in closely related areas (e.g., process verification [10] and automated planning [21]), to the best of our knowledge, this is the first work that explicitly addresses the two types of compliance from a fully declarative point of view, and proposes a formal framework for modelling and reasoning with them. The paper identifies further areas of research –in the *compliance by design* space– stemming from the work presented here, namely: process compliance resolution (how to revise a non-compliant theory) and process derivation (how to extract a business process from compliant declarative specifications for it). We have outlined some possible developments inspired by automated planning [21] and process mining [20].

The closest work to our approach is [22] proposing LTL (Linear Temporal Logic) to describe compliance rules, to use automated reasoning techniques to generate LTL models of processes and then using process mining techniques to extract business processes. The main limitations of this work is that, while LTL is suitable to represents the temporal relationships involving the tasks, alone is not suitable to faithfully represent the normative (nor business) requirements.

Defeasible Logics are a very powerful tool to describe an environment, and in the years scholars extended the primitive formalism to deal with many different kinds of situations. [23] introduces a temporalised DL, while [13] describes many types of obligations, and shows how control flows (and other relationships among process tasks) are modelled using the various types of obligations. It seems very likely that the formalism introduced in this work can be further extended to handle both temporal constraints, and different types of obligations either when determining if a Modal DL theory is compliant with sets of norms and goals, revising it, or creating a business process starting from its compliance. It seems also of great interest to incorporate idea from [24] to model resources and complex events in our logical framework.

**Acknowledgements.** NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy, the Australian Research Council through the ICT Centre of Excellence program and the Queensland Government.

## References

1. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - R&D 23, 99–113 (2009)
2. Governatori, G., Sadiq, S.: The journey to business process compliance. In: Handbook of Research on BPM, pp. 426–454 (2008)
3. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Transactions on Computational Logic 2, 255–287 (2001)
4. Governatori, G.: Representing business contracts in RuleML. International Journal of Cooperative Information Systems 14, 181–216 (2005)
5. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: Hung, P.C.K. (ed.) 10th International Enterprise Distributed Object Computing Conference (EDOC 2006), pp. 221–232. IEEE Computing Society (2006)
6. Sadiq, S., Governatori, G., Naimiri, K.: Modelling of control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
7. Governatori, G., Rotolo, A.: Bio logical agents: Norms, beliefs, intentions in defeasible logic. Journal of Autonomous Agents and Multi Agent Systems 17, 36–69 (2008)
8. Governatori, G., Rotolo, A.: Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations. Australasian Journal of Logic 4, 193–215 (2006)
9. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Trans. Comput. Logic 2, 255–287 (2001)
10. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers 8, 21–66 (1998)
11. Dastani, M., Governatori, G., Rotolo, A., van der Torre, L.: Programming cognitive agents in defeasible logic. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 621–636. Springer, Heidelberg (2005)
12. Dastani, M., Governatori, G., Rotolo, A., van der Torre, L.: Preferences of agents in defeasible logic. In: Zhang, S., Jarvis, R.A. (eds.) AI 2005. LNCS (LNAI), vol. 3809, pp. 695–704. Springer, Heidelberg (2005)
13. Governatori, G., Rotolo, A.: Norm compliance in business process modeling. In: [25], pp. 194–209
14. Ghose, A., Koliadis, G.: Auditing Business Process Compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
15. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. 36, 498–516 (2011)
16. Schumm, D., Leymann, F., Ma, Z., Scheibler, T., Strauch, S.: Integrating Compliance into Business Processes Process Fragments as Reusable Compliance Controls. Universitätsverlag Göttingen, 2125–2137 (2010)

17. Schumm, D., Türetken, O., Kokash, N., Elgammal, A., Leymann, F., van den Heuvel, W.J.: Business Process Compliance Through Reusable Units of Compliant Processes. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 325–337. Springer, Heidelberg (2010)
18. Governatori, G., Olivieri, F., Scannapieco, S., Cristani, M.: Superiority based revision of defeasible theories. In: [25], pp. 104–118
19. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
20. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16, 1128–1142 (2004)
21. Ghallab, M., Nau, D.S., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann (2004)
22. Awad, A., Goré, R., Thomson, J., Weidlich, M.: An Iterative Approach for Business Process Template Synthesis from Compliance Rules. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 406–421. Springer, Heidelberg (2011)
23. Governatori, G., Rotolo, A., Sartor, G.: Temporalised normative positions in defeasible logic. In: ICAIL 2005, pp. 25–34. ACM Press (2005)
24. Governatori, G., Rotolo, A., Sadiq, S.: A model of dynamic resource allocation in workflow systems. In: ADC 2004, pp. 197–206. ACS (2004)
25. Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.): RuleML 2010. LNCS, vol. 6403. Springer, Heidelberg (2010)