The Journey to Business Process Compliance

Guido Governatori
NICTA, Queensland Research Laboratory
Brisbane, QLD, Australia
guido.governatori@nicta.com.au

Shazia Sadiq
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD, Australia
shazia@itee.uq.edu.au

1 October 2008

Abstract

It is a typical scenario that many organisations have their business processes specified independently of their business obligations (which includes contractual obligations to business partners, as well as obligations a business has to fulfil against regulations and industry standards). This is because of the lack of guidelines and tools that facilitate derivation of processes from contracts but also because of the traditional mindset of treating contracts separately from business processes. This chapter will provide a solution to one specific problem that arises from this situation, namely the lack of mechanisms to check whether business processes are compliant with business contracts. The chapter begins by defining the space for business process compliance and the eco-system for ensuring that process are compliant. The key point is that compliance is a relationship between two sets of specifications: the specifications for executing a business process and the specifications regulating a business. The central part of the chapter focuses on a logic based formalism for describing both the semantics of normative specifications and the semantics of compliance checking procedures.

1 Introduction

The term compliance is applied in many disciplines such as management, standards development, regulations, medical practice and so on. It is often used to denote and demonstrate adherence of one set of rules (we refer to them as 'source rules' hereafter) against other set of rules (we refer to them as 'target rules' hereafter). Typically, target rules represent an established or agreed set of guidelines, norms, laws, regulations, recommendations or qualities which, if obeyed, will deliver certain effect or value to those to whom they can apply, or to those with whom they interact. In some way, the target rules are intended for a global or broad community of participants in a specific universe of discourse. On the other hand, source rules are developed to apply to participants and their behaviours in certain local contexts, and adherence of source rules to the target rules then ensures that both local and global expectations or requirements can be met.

In management for example, target rules represent policies that need to be obeyed by companies, their staff or executives, while undertaking their normal course of actions to meet their goals. Examples of such rules are the US regulations such as Sarbanes-Oxley Act[1] or Health Insurance Privacy Act (HIPPA)[2]. In standards development, compliance requirements are stated to ensure necessary consistency of one set of requirements with some broader set of requirements, e.g., a compliance of the ODP Enterprise Language with ODP-RM[3]. Note that in standards communities, the term conformance has a different meaning: it is used to relate an implementation to a standard specification. Finally, in health sector, compliance is referred to a patient's (or doctor's) adherence to a recommended course of treatment.

Similarly, we apply this interpretation of compliance as a metaphor to discuss adherence or consistence of a set of rules in business processes against a set of rules regulating a particular business. This set of rules can stem from different sources, legislation, standards, best practices, internal guidelines and policies, contracts between the parties involved in the process and so on. We will refer to the source of these as normative documents, and to the rules themselves as norms or normative specifications. So, ensuring compliance of business processes with a normative document means ensuring consistency of norms stated in normative documents and rules covering the execution of business processes. In other words, to check that the specification of a business process complies with a normative document regulating the domain of the process, one has to verify that all execution paths of the process, possible according to the specification of the business process, comply with the normative specification. This means that no execution path is in breach of the regulation. This consistency, for example, is necessary to satisfy commitments that parties typically state in their agreements or business contracts while carrying out their mutually related internal business activities. Such compliance also leads to benefits to both parties, e.g., minimisation of costs or damages to either party whether these are associated with potentially inadvertent behaviour or deliberate violations while seeking more opportunistic engagements.

1.1 Compliance Space

Compliance of business processes with normative documents is thus important to ensure establishing better links between these two traditionally separate universes of discourse, i.e., legal and business process spaces (see Figure 1).

Firstly, the source of the normative specifications and the business specifications (i.e., the design of the process to meet the objectives of a business) will be distinct both from an ownership and governance perspective, as well as from a timeline perspective. Where as businesses can be expected to have some form of business objectives, normative spec-

[1]Sarbanes-Oxley Act of 2002, US Public Law 107-204.

[2]Health Insurance Portability and Accountability Act of 1999. US Public Law 104-191.

[3]ITU-T Rec X.902, ISO/IEC 10746-2: Foundations, RM-ODP

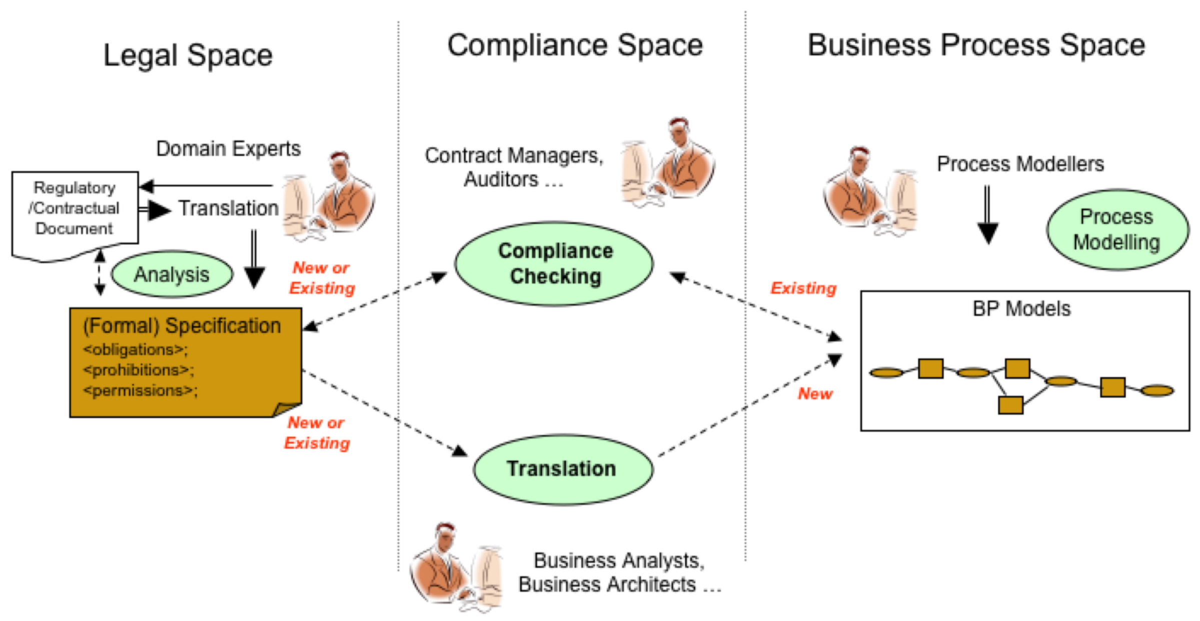


Figure 1: Compliance Space

ifications will be dictated by mostly external sources and often the various norms regulating a business are created at different times and are subject to evolution over time to accommodate changes in the business and in the society.

Secondly, the two have differing concerns, namely business objectives and normative objectives. Thus the use of business process languages to model normative specifications may not provide a conceptually faithful representation. The focus in the legal space is to describe what processes have to do, what should be avoided in the execution of a process. Thus the major concern in the legal space is on *what* a business has to do. Accordingly, in this space we have a declarative perspective of the objectives of processes, indicating what needs to be done (in order to comply). The business process space, on the other hand, has been the focus of management science, such as various business process reengineering approaches. This is a domain of business process modellers and business architects involved in enterprise architecture developments. These professionals have typically been involved in identifying business requirements and then designing business processes to satisfy these requirements. Accordingly, business process specifications are fundamentally prescriptive in nature, i.e., detailing how business activities should take place. There is evidence of some developments towards descriptive approaches for BPM, but these works were predominantly focused on achieving flexibility in business process execution, see e.g. (Pesic and van der Aalst, 2006; Sadiq et al., 2005).

Thirdly, there is likelihood of conflicts, inconsistencies and redundancies within the two specifications. Thus the intersection of the two needs to be carefully studied. This is where the compliance space plays its role. The compliance space however, is a new area of interest and endeavour, in particular driven by recent regulative and legislative acts, which require the establishment of stronger and more enforceable compliance requirements against the target set of rules. Some of the largest scandals in corporate history, such as Enron, have led to an increased importance of compliance and related initiatives within organisations. Therefore this new space has led to the development of new roles such as compliance auditors, or requirements for new skills to be developed by existing roles, such as contract managers, business analysts or business architects, for the contract/compliance management domain.

1.2 Managing Compliance

Ensuring compliance of business processes with normative documents is a complex problem involving a number of alternatives. Currently there are two main approaches towards achieving compliance. The first one is *retrospective reporting*, wherein traditional audits are conducted for "after-the-fact" detection, often through manual checks by expensive consultants. With increasing pressures and penalties for non-compliance, this approach is rather limited.

A second and more recent approach is to provide some level of automation through *automated detection*. The bulk of existing software solutions for compliance follow this approach. The proposed solutions hook into variety of enterprise system components (e.g. SAP HR, LDAP Directory, Groupware etc.) and generate audit reports against hard-coded checks performed on the system. These solutions often specialise in certain class of checks, for example the widely supported checks that relate to Segregation of Duty violations in role management and user provisioning systems. Such monitoring capability assists in checking for compliance against the hard-coded checks and consequently in the remediation and/or mitigation of control deficiencies. However, this approach still resides in the space of "after-the-fact" detection.

We believe that a sustainable approach for achieving compliance should fundamentally have a *preventative* focus. As such, we describe an approach that provides the capability to capture compliance requirements through a generic requirements modelling framework, and subsequently facilitate the propagation of these requirements into business process models and enterprise applications, thus achieving *compliance by design*.

The approach we describe in this chapter can be applied to problems common to many enterprises, i.e., there are many existing processes that were designed in the absence of any knowledge of specific regulations, opening possibilities for violations. This requires checking compliance of processes against norms. This can be either against the existing norms, to fix possible inconsistencies that have not been detected yet, or against new regulations to detect whether existing business processes can lead to conflicts with new norms, either in terms of incompatible rules or in terms of unrealistic resource expectations that new legislation may require (see Figure 1).

1.3 Organisation of the Chapter

In this chapter we describe an approach to business process compliance based on (semantic) annotations, where the annotations are written in the formal language chosen to represent the normative specifications. The idea is that business processes are annotated and the annotations provide the conditions a process has to comply with. Annotations can be defined at different levels. For example, we can annotate a full process or a single task in a process. In addition, we can have different types of annotations. Annotations can range from the full set of rules (norms) specific to a process or a single task to simple semantic annotation corresponding to one effect of a particular task, e.g., after the successful execution of task A in a process B the value of the environment variable C is D.

In order to support the above technique based on annotations we first need a formal representation of normative specifications. We will address this issue in the next section where we describe a formalism able to capture the notions need for the representation of normative specifications.

2 Normative Specifications

Compliance is a relationship between two sets of specifications: the normative specifications that prescribe what a business has to do, and the process modelling specification describing how a business performs its activities. Accordingly to properly verify that a process/procedure complies with the norms regulating the particular business one has to provide conceptually sound representations of the process on one side and the norms on the other, and then check the alignment of the formal specifications of the process and the formal specifications for the norms. This means that the normative specifications tell us what obligations, permissions, prohibitions a process is subject to. A normative document often contains many norms regulating a business. Thus a normative document can be seen as a normative system. Normative systems can be modelled with the help of Deontic Logic. Deontic Logic is the branch of logic that studies the formal properties of normative notions (also called normative positions) such as obligations, permissions, prohibitions. In particular, Deontic Logic can be used to investigate the mutual relationships among the various normative positions, how complex normative positions (e.g., delegation, empowerment, rights and so) can be expressed using simpler one, and the relationships between the norms in a normative system.

Standard Deontic Logic (SDL) is a starting point for logical investigation of the basic normative notions and it offers a very idealised and abstract conceptual representation of these notions but at the same time it suffers from several drawbacks given its high level of abstraction (Sartor, 2005). Over the years many different deontic logics have been proposed to capture the different intuitions behind these normative notions and to overcome drawbacks and limitations of SDL. One of the main limitations in this context is its inability to reason with violations, and the obligations arising in response to violations (Carmo and Jones, 2002). Very often normative statements pertinent to business processes, and in particular contracts, specify conditions about when other conditions in the document have nor been fulfilled, that is when some (contractual) clauses have been violated. Hence, any formal representation, to be conceptually faithful, has to been able to deal with this kind of situations.

In the rest of the section we introduce the basic notions of Deontic Logic and then we present a particular deontic logic that addresses the issue discussed above and that is suitable for checking compliance of business processes.

2.1 Formalising Deontic Constraints

Deontic logic extends first order logic with the deontic operators O, P and F denoting obligations, permissions and prohibitions. The deontic operators satisfy the following equivalence relations:

$$OA \equiv \neg P \neg A \quad \neg O \neg A \equiv PA \quad O \neg A \equiv FA \quad \neg PA \equiv FA.$$

The operators also satisfy the following relationship $OA \rightarrow PA$, meaning that if A is obligatory, then A is permitted. This relationship can be used to ensure checking of the internal consistency of the obligations in a set of norms it is possible to execute obligations without doing something that is forbidden. We extend the notation to cover the subject to whom a normative position applies to. In case of obligation, this can be denoted using the expression $O_s A$ to be read as 's has the obligation to do A', or 'A is obligatory for s'. Where A represents a factual statement. Thus, for example, where A is the proposition 'payTaxes', $O_s A$ means that "s has the obligation to pay the taxes" or that 'paying the taxes is obligatory for s'. Similarly for the other operators.

In case of certain breaches of norms, special norms/policies may be included to express the respective obligations for the actors involved in a process. These policies can vary from pecuniary penalties to the termination of a contract and so on. In deontic logic, this type of expression, namely the activation of certain obligations in case of other obligations being violated, is referred to as contrary-to-duty obligations (CTD) or reparation obligations (because they are intended to 'repair' or 'compensate' violations of primary obligations). The reparation obligations are in force only when normative violations occur and are meant to 'repair' violations of primary obligations. Thus a reparation policy is a conditional obligation arising in response to a violation, where a violation is signalled by an unfulfilled obligation. The expression of violation conditions and the reparation obligations is an important requirement for formalising norms, design subsequent business processes to minimise or deal with such violations and also to determine the compliance of a process with the relevant norms.

There are a number of different approaches in deontic logic to formalise CTD obligations, but in this paper we use a simple logic of violation, to avoid danger of logical paradoxes that some other approaches may involve (Carmo and Jones, 2002). This logic is also suitable to model chains of violations as described next.

2.2 Formalising violations of deontic constraints

In addition to using the logic based approach to specifying core deontic constraints, we thus provide a simple logic of violation.

The violation expression consists of the primary obligation, its violation conditions, an obligation generated upon the violation condition occurs, and this can recursively be iterated, until the final condition is reached. We introduce the non-boolean connective $\otimes$, whose interpretation is such that $OA \otimes OB$ is read as "OB is the reparation of the violation of OA". In other words the interpretation of $OA \otimes OB$, is that A is obligatory, but if the obligation OA is not fulfilled (i.e., the obligation expressed by OA is violated, i.e., we have $\neg A$), then the obligation OB is activated and becomes in force until it is satisfied or violated. If $OA \otimes OB$ appears in a longer chain of obligations/reparations, e.g., $OA \otimes OB \otimes OC$, then the violation of the OB activate a new obligation, i.e., OC. Similarly for longer chains.

2.3 Formal Contract Logic (FCL)

We now provide a formal account of the idea presented in Section 2.2 which we will refer to as Formal Contract Logic (FCL). FCL was introduced in (Governatori, 2005) for the formal analysis of business contracts. FCL is a combination of an efficient non-monotonic formalism (defeasible logic (Antoniou et al., 2001, 2006)) and a deontic logic of violations (Governatori and Rotolo, 2006). This particular combination allows us to represent exceptions as well as the ability to capture violations, the obligations resulting from the violations, and the reparations. In addition FCL has good computational properties: the extension of a theory (i.e., the set of conclusions/normative positions following from a set of facts can be computed in time linear to the size of the theory).

The ability to handle violation is very important for compliance of business processes. Often business processes are deployed in dynamic and somehow unpredictable environments. As a consequence, in some cases, maybe due to external circumstances, it is not possible to operate in the way specified by the norms, but the norms prescribe how to recover from the resulting violations. In other cases, the prescribed behaviour is subject to exceptions. Finally, in other cases, one might not have a complete description of the environment. Accordingly the process has to operate based on the available input, but if more information was available, then the task to be performed could be a different one (this is typically the case of the *due diligence* conditions, where one has to act in a 'reasonable' way based on the available information, but if a more complete description the behaviour might be different). A conceptually sound formalisation of norms (for assessing the compliance of a process) should take into account all the aspects mentioned above. FCL is sound in this respect given the combinations of the deontic component (able to represent the fundamental normative positions and chains of violations/reparations) and the defeasible component that takes care of the issue about partial information and possibly conflicting provisions.

The language of FCL consists of the following set of atomic symbols: a numerable set of propositional letters $p, q, r, \ldots$, intended to represent the state variables and the tasks of a process. Formulas of the logic are constructed using the deontic operators O (for obligation), P (for permission), negation $\neg$ and the non-boolean connective $\otimes$ (for the contrary-to-duty operator). The formulas of FCL will be constructed in two steps according to the following formation rules:

- every propositional letter is a literal;
- the negation of a literal is a literal;
- if X is a deontic operator and l is a literal then Xl and $\neg Xl$ are deontic literals.

After we have defined the notions of literal and deontic literal we can use the following set of formation rules to introduce $\otimes$-expressions, i.e., the formulas used to encode chains of obligations and violations.

- every deontic literal is an $\otimes$-expression;
- if $Ol_1, \ldots, Ol_n$ are deontic literals and l_{n+1} is a literal, then $Ol_1 \otimes \ldots \otimes Ol_n$ and $Ol_1 \otimes \ldots \otimes Ol_n \otimes Pl_{n+1}$ are $\otimes$-expressions (we also refer to $\otimes$-expressions as obligation chains or simply chains).

The connective $\otimes$ permits combining primary and contrary-to-duty obligations into unique regulations. The meaning of an expression like $O_sA \otimes O_sB \otimes O_sC$ is that the primary obligation for s is A, but if A is not done, then s has the obligation to do B. But if event B fails to be realised, then s has the obligation to do C. Thus B is the reparation of the violation of the obligation O_sA (to have violation of an obligation such as O_sA we must have that A does not hold; this mean that the negation of A, i.e., $\neg A$, holds). Similarly C is the reparation of the obligation O_sB, which is force when the violation of A occurs.

The formation rules for $\otimes$-expressions allow a permission to occur only at the end of such expressions. This is due to the fact that a permission can be used as a reparation of a violation, but it is not possible to violate a permission, thus it makes no sense to have reparations to permissions.

Each condition or norm of a normative document is represented by a rule in FCL, where a rule is an expression

$$r : A_1, \dots, A_n \Rightarrow C$$

where r is the name/id of the norm, $A_1, \dots, A_n$ –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) and C is the conclusion of the rule. Each A_i is either a literal or a deontic literal and C is an $\otimes$-expression.

The meaning of a rule is that the normative position (obligation, permission, prohibition) represented by the conclusion of the rule is in force when all the premises of the rule hold. Thus, for example, suppose we have a contract for the provision for a service where we have the following clause:

> "**5.1** the supplier (S) shall refund the purchaser (P) and pay a penalty of \$1000 in case she does not replace within 3 days a service that does not conform with the published standards"

This clause can be represented as:[4]

$$r : \neg a, \neg b \Rightarrow O_S c$$

where the propositional letter a means "a service has been provided according to the published standards", b stands for the event "replacement occurred within 3 days", and c represents the event "refund the customer and pay her the penalty". The norm is activated, i.e., the supplier is obliged to refund the customer and pay a penalty of \$1000, when the condition $\neg a$ is true (i.e., we have a faulty service), and the event "replacement occurred within 3 days" lapsed, i.e., its negation ($\neg b$) occurred.

FCL is equipped with rule set. The superiority relation ($\prec$) determines the relative strength of two rules, and it is used when rules have potentially conflicting conclusions. For example given the rule $r_1 : A \Rightarrow B \otimes C$ and $r_2 : D \Rightarrow \neg C$. $r_1 \prec r_2$ means that rule r_1 prevails over rule r_2 in situation where both fire and they are in conflict (i.e., rule r_2 fires for the secondary obligation C).

[4]In the remaining of the chapter we will use O_S and P_S for the obligation and permission operators relative to the *Supplier*, and O_P and P_P for the *Purchaser*. O_s and P_s will be used for a generic subject.

2.4 Normal Forms

We introduce transformations of an FCL representation of a normative document to produce a normal form of the same (NFCL). A normal form is a representation of a normative document based on an FCL specification containing all conditions that can generated/derived from the given FCL specification. The purpose of a normal form is to "clean up" the FCL representation of a normative document, that is to identify formal loopholes, deadlocks and inconsistencies in it, and to make hidden conditions explicit.

In the rest of this section we introduce the procedures to generate normal forms. First (Section 2.4.1) we describe a mechanism to derive new contract conditions by merging together existing normative clauses. In particular we link an obligation and the obligations triggered in response to violations of the obligation. Then, in Section 2.4.2, we examine the problem of redundancies, and we give a condition to identify and remove redundancies from the formal normative specification.

2.4.1 Merging Norms

One of the features of the logic of violations is to take two rules, or norms, and merge them into a new clause. In what follows we will first examine some common patterns of this kind of construction and then we will show how to generalise them.

Let us consider a norm like (in what follows Γ and Δ are sets of premises)

$$\Gamma \Rightarrow O_s A.$$

Given an obligation like this, if we have that the violation of $O_s A$ is part of the premises of another norm, for example,

$$\Delta, \neg A \Rightarrow O_{s'} C,$$

then the latter must be a good candidate as reparational obligation of the former. This idea is formalised is as follows:

$$\frac{\Gamma \Rightarrow O_s A \qquad \Delta, \neg A \Rightarrow O_{s'} C}{\Gamma, \Delta \Rightarrow O_s A \otimes O_{s'} C}$$

This reads as follows: given two policies such that one is a conditional obligation ($\Gamma \Rightarrow O_s A$) and the antecedent of second contains the negation of the propositional content of the consequent of the first ($\Delta, \neg A \Rightarrow O_{s'} C$), then the latter is a reparational obligation of the former. Their reciprocal interplay makes them two related norms so that they cannot be viewed anymore as independent obligations. Therefore we can combine them to obtain an expression (i.e., $\Gamma, \Delta \Rightarrow O_s A \otimes O_{s'} C$) that exhibits the *explicit reparational obligation* of the second norm with respect to the first. Notice that the subject of the primary obligation and the subject of its reparation can be different, even if very often they are the same.

Suppose that a contract includes the rules

$$r : \mathit{Invoice} \Rightarrow O_P \mathit{PayWithin7Days}$$
$$r' : \neg \mathit{PayWithin7Days} \Rightarrow O_P \mathit{PayWithInterest}.$$

From these we obtain

$$r'' : \mathit{Invoice} \Rightarrow O_P \mathit{PayWithin7Days} \otimes O_P \mathit{PayWithInterest}.$$

We can also generate chains of CTDs in order to deal iteratively with violations of reparational obligations. The following case is just an example of this process.

$$\frac{\Gamma \Rightarrow O_s A \otimes O_s B \qquad \neg A, \neg B \Rightarrow O_s C}{\Gamma \Rightarrow O_s A \otimes O_s B \otimes O_s C}$$

For example we can consider the situation described by the hypothetical clause 5.1 discussed at page 8, whose formal representation is given by the rules

$$r : \mathit{Invoice} \Rightarrow O_S \mathit{QualityOfService} \otimes O_S \mathit{Replace3days}$$
$$r' : \neg \mathit{QualityOfService}, \neg \mathit{Replace3days} \Rightarrow O_S \mathit{Refund\&Penalty}$$

from which we derive the new rule

$$r'' : \mathit{Invoice} \Rightarrow O_S \mathit{QualityOfService} \otimes O_S \mathit{Replace3days} \otimes O_S \mathit{Refund\&Penalty}.$$

The above patterns are just special instances of the general mechanism described in details in (Governatori and Rotolo, 2006; Governatori, 2005).

2.4.2 Removing Redundancies

Given the structure of the inference mechanism it is possible to combine rules in slightly different ways, and in some cases the meaning of the rules resulting from such operations is already covered by other rules in the contract. In other cases, the rules resulting from the merging operation are generalisations of the rules used to produce them, consequently, the original rules are no longer needed in the specifications. To deal with this issue we introduce the notion of subsumption between rules. A rule subsumes a second rule when the behaviour of the second rule is implied by the first rule.

We first introduce the idea with the help of some examples and then we show how to give a formal definition of the notion of subsumption appropriate for FCL.

Let us consider the rules:

$$r : \mathit{Service} \Rightarrow O_S \mathit{QualityOfService} \otimes O_S \mathit{Replace3days} \otimes O_S \mathit{Refund\&Penalty},$$
$$r' : \mathit{Service} \Rightarrow O_S \mathit{QualityOfService} \otimes O_S \mathit{Replace3days}.$$

The first rule, r, subsumes the second r'. Both rules state that after the supplier has provided the service she has the obligation to provide the service according to the published standards, if she violates such an obligation, then the violation of *QualityOfService* can be repaired by replacing the faulty service within three days ($O_S \mathit{Replace3days}$). In other words, $O_S \mathit{Replace3days}$ is a secondary obligation arising from the violation of the primary obligation $O_S \mathit{QualityOfService}$. In addition r prescribes that the violation of the secondary obligation $O_S \mathit{Replace3days}$ can be repaired by $O_S \mathit{Refund\&Penalty}$, i.e., the seller has to refund the buyer and in addition she has to pay a penalty.

As we discussed in the previous paragraphs, the conditions of a normative document cannot be taken in isolation in so far as they exist in the document. Consequently, the whole normative document determines the meaning of each single clause (norm) in it. In agreement with this holistic view of norms we have that the normative content of r' is included in that of r. Accordingly, r' does not add any new piece of information to the contract, it is redundant and can be dispensed from the explicit formulation of the norms.

Another common case is exemplified by the rules:

$$r : Invoice \Rightarrow O_P PayWithin7Days \otimes O_P PayWithInterest$$
$$r' : Invoice, \neg PayWithin7Days \Rightarrow O_P PayWithInterest.$$

The first rule says that after the seller sends the invoice the buyer has one week to pay, otherwise the buyer has to pay the principal plus the interest. Thus, we have the primary obligation $O_P PayWithin7Days$, whose violation is repaired by the secondary obligation $O_P PayWithInterest$, while, according to the second rule, given the same set of circumstances $Invoice$ and $\neg PayWithin7Days$ we have the primary obligation $O_P PayWithInterest$. However, the primary obligation of r' obtains when we have a violation of the primary obligation of r. Thus, the condition of applicability of the second rule includes that of the first rule, which then is more general than the second and we can discard r' from the formal representation of the specifications.

The intuitions we have just exemplified is captured by the following definition.

Definition 1 *Let* $r_1 : \Gamma \Rightarrow A \otimes B \otimes C$ *and* $r_2 : \Delta \Rightarrow D$ *be two rules, where* $A = \bigotimes_{i=1}^{m} A_i$*,* $B = \bigotimes_{i=1}^{n} B_i$ *and* $C = \bigotimes_{i=1}^{p} C_i$*. Then* r_1 subsumes r_2 *iff*

1. $\Gamma = \Delta$ *and* $D = A$*; or*
2. $\Gamma \cup \{\neg A_1, \dots, \neg A_m\} = \Delta$ *and* $D = B$*; or*
3. $\Gamma \cup \{\neg B_1, \dots, \neg B_n\} = \Delta$ *and* $D = A \otimes \bigotimes_{i=0}^{k \le p} C_i$*.*

The intuition is that the normative content of r_2 is fully included in r_1. Thus, r_2 does not add anything new to the system and it can be safely discarded.

Conflicts often arise in normative systems. What we have to determine is whether we have genuine conflicts, i.e., the norms are in some way flawed or whether we have *prima-facie* conflicts. A prima-facie conflict is an apparent conflict that can be resolved when we consider it in the context where it occurs and if we add more information the conflict disappears. For example let us consider the following two rules:

$$r : PremiumCustomer \Rightarrow O_S Discount$$
$$r' : SpecialOrder \Rightarrow O_S \neg Discount$$

saying that premium customers are entitled to a discount (r), but there is no discount for goods bought with a special order (r'). Is a premium customer entitled to a discount when she places a special order? If we only have the two rules above there is no way to solve the conflict just using the contract and there is the need of a domain expert to advise the knowledge engineer about what to do in such case.[5] The logic can only point out that there is a conflict in the contract. On the other hand, if we have an additional provision

$$r'' : PremiumCustomer, \neg Discount \Rightarrow O_s Rebate$$

[5]As we have seen in Section 2.3 FCL has a superiority relation over rules to handle situations like this one.

Specifying that if for some reasons a premium customer did not received a discount then the customer is entitled to a rebate on the next order, then it is possible to solve the conflict, because the contract allows a violation of rule r to be amended by r''.

We can now introduce the mechanism for making explicit conflicting norms (contradictory norms) within the system:

$$\frac{\Gamma \Rightarrow A \qquad \Delta \Rightarrow \neg A}{\Gamma, \Delta \Rightarrow \bot}$$

where

1. there is no rule $\Gamma' \Rightarrow X$ such that either $\neg A \in \Gamma'$ or $X = A \otimes B$;
2. there is no conditional rules $\Delta' \Rightarrow X$ such that either $A \in \Delta'$ or $X = \neg A \otimes B$;
3. for any formula B, $\{B, \neg B\} \not\subseteq \Gamma \cup \Delta$.

The meaning of these three conditions is that given two rules, we have a conflict if the normative content of the two rules is opposite, such that none of them can be repaired, and the states of affairs/preconditions they require are consistent.

Once conflicts have been detected there are several ways to deal with them. The first thing to do is to determine whether we have a *prima-facie* conflict or a genuine conflict. As we have seen we have a conflict when we have two rules with opposite conclusions. Thus, a possible way to solve the conflict is to create a superiority relation over the rules and to use it do "defeat" the weaker rule. In Section 2.5 we will examine how to reason with norms, and we will see how to use the superiority relation to solve conflicts.

2.4.3 Normalisation Process

We now describe how to use the machinery presented in Section 2.4.1 and Section 2.4.2 to obtain FCL normal forms. The FCL normal form of a normative document provides a logical representation of normative specifications in format that can be used to check the compliance of a process. This consists of the following three steps:

1. Starting from a formal representation of the explicit clauses of a set of normative specifications we generate all the implicit conditions that can be derived from the normative document by applying the merging mechanism of FCL.
2. We can clean the resulting representation of the contract by removing all redundant rules according to the notion of subsumption.
3. Finally, we use the conflict identification rule to label and detect conflicts.

In general, the process at step 2 must be done several times in the appropriate order as described above. The normal form of a set of rules in FCL is the fixed-point of the above constructions. A normative document contains only finitely many rules and each rule has finitely many elements. In addition it is possible to show that the operation on which the construction is defined is monotonic (Governatori and Rotolo, 2006), thus according to standard set theory results the fixed-point exists and it is unique. Notice that the computation of the fixed-point depends on the order in which the operations of subsumption and merging are performed (subsumption fixed and merging after). Changing the order of these operations, i.e. merging first and subsumption after, or interleaving the two operations does not produce the same result. Specifically, some rules might be excluded from the computation.

2.5 Reasoning with Norms

In the previous section we have examined the mechanism to obtain a set of rules covering all possible (explicit) norms for obligations, permissions and prohibitions that can arise from an initial set of norms. In this section we focus on the issue of how to determine what obligations are in force for a specific situations. Thus taking the well known distinction between schema and instance. The previous section defines the procedure to obtain the full (normalised) schema corresponding to a normative document. Here we study how to get the normative positions active for a specific instance of a business process. The reasoning mechanism of FCL is an extension of Defeasible Logic.

Defeasible logic, originally created by Donald Nute (Nute 1994) with a particular concern about efficiency and implementation, is a simple and efficient rule based non-monotonic formalism. Over the year, the logic has been developed and extended, and several variants have been proposed to model different aspects of normative reasoning and encompassed other formalisms to for normative reasoning.

The main intuition of the logic is to be able to derive "plausible" conclusions from partial and sometimes conflicting information. Conclusions are *tentative* conclusions in the sense that a conclusion can be withdrawn when we have new pieces of information.

The knowledge in a Defeasible Theory is organised in *facts* and *rules* and *superiority relation*.

- Facts are indisputable statements.
- Defeasible rules are rules that can be defeated by contrary evidence.
- The superiority relation in a binary relation defined over the set of rules. The superiority relation determines the relative strength of two (conflicting) rules.

The meaning of a defeasible rule, like

$$A_1,\dots,A_n \Rightarrow C$$

is that normally we are allowed to derive C given $A_1,\dots,A_n$, unless we have some reasons to support the opposite conclusion (i.e., we have a rule like $B_1,\dots,B_m \Rightarrow \neg C$).

Defeasible Logic is a "skeptical" non-monotonic logic, meaning that it does not support contradictory conclusions. Instead, Defeasible Logic seeks to resolve conflicts. In cases where there is some support for concluding A but also support for concluding $\neg A$, Defeasible Logic does not conclude either of them (thus the name skeptical). If the support for A has priority over the support for $\neg A$ then A is concluded.

A defeasible conclusion is a tentative conclusion that might be withdrawn by new pieces of information, or in other terms it is the 'best' conclusion we can reach with the given information. In addition, the logic is able to tell whether a conclusion is or is not provable. Thus, it is possible to have the following 2 types of conclusions:

- Positive defeasible conclusions: meaning that the conclusions can be defeasible proved;
- Negative defeasible conclusions: meaning that one can show that the conclusion is not even defeasibly provable.

A defeasible conclusion A can be derived if there is a rule whose conclusion is A, whose prerequisites (antecedent) have either already been proved or given in the case at hand (i.e., facts), and any stronger rule whose conclusion is $\neg A$ (the negation of A) has prerequisites that fail to be derived. In other words, a conclusion A is (defeasibly) derivable when:

1. A is a fact; or
2. there is an applicable defeasible rule for A, and either
   (a) all the rules for $\neg A$ are discarded (i.e., not applicable) or
   (b) every applicable rule for $\neg A$ is weaker than an applicable strict or defeasible rule for A.

A rule is applicable if all elements in the body of the rule are derivable (i.e., all the premises are positively provable), and a rule is discarded if at least one of the element of the body is not provable (or it is a negative defeasible conclusion).

2.5.1 Defeasible Logic at Work

We illustrate the inferential mechanism of Defeasible Logic with the help of an example. Let us assume we have a theory containing the following rules:

$$r_1 : \mathit{PremiumCustomer}(X) \Rightarrow \mathit{Discount}(X)$$
$$r_2 : \mathit{SpecialOrder}(X) \Rightarrow \neg\mathit{Discount}(X)$$
$$r_3 : \mathit{Promotion}(X) \Rightarrow \neg\mathit{Discount}(X)$$

where the superiority relation is thus defined: $r_1 \prec r_3$ and $r_2 \prec r_1$. The theory states that services in promotion are not discounted, and so are special orders with the exception of special orders placed by premium customers, who are normally entitled to a discount.

Consider a scenario where the only piece of information available is that we have received a special order. In this case we can conclude that the price has to be calculated without a discount since rule r_1 is not applicable (we do not know whether the customer is a premium customer or not). In case the special order is received from a premium customer for a service not in promotion, we can derive that the customer is entitled to a discount. Indeed rule r_1 is now applicable and it is stronger than rule r_2, and r_3, which is stronger than r_2 is not applicable (i.e., the service is not in promotion).

2.5.2 Adding Reparation Chains

FCL is an extension of defeasible logic with the reparation operator ($\otimes$). Accordingly the reasoning mechanism to derive conclusion is an extension of that for defeasible logic. In defeasible logic the conclusions of a rule is a single literal and not a reparation chain. Thus, the condition that OA appears in the conclusion of a rule means in defeasible logic that OA is the conclusions of the rule. FCL extends defeasible logic with reparation chains, thus, we have to extend the reasoning mechanism of defeasible logic to accommodate the additional construction provided by FCL. To prove OA, we have to consider all rules with a reparation chain for OA, where for all elements before OA in the chain, the negation of the element is already provable. Thus to prove A given a rule

$$P_1,\dots,P_n \Rightarrow OC_1 \otimes \cdots \otimes OC_m \otimes OA \otimes OD_1 \otimes \cdots \otimes OD_k,$$

we have that $P_1,\dots,P_n$ must be all provable, and so must be $\neg C_1,\dots,\neg C_m$. For the full details see (Governatori, 2005).

Consider a process governed by the following rule

$$r : \mathit{Invoice} \Rightarrow O_P\mathit{PayWithin7Days} \otimes O_P\mathit{PayWithInterest}$$

and a situation where an invoice has been received. Thus, we have *Invoice*. According to the rule we derive that the obligation in force is $O_P\mathit{PayWithin7Days}$. Suppose now that, for some reasons, the invoice has not been paid until the tenth day after the reception. The facts of the case at hand are *Invoice* and $\neg\mathit{PayWithin7Days}$. Here, according to the rule and the reasoning mechanism of FCL, we obtain that the obligation in force is $O_P\mathit{PayWithInterest}$.

2.6 Summary

In this section we have argued about the need of conceptual model of normative specifications and we have illustrated how extensions of (Standard) Deontic Logic provide sound conceptual specifications suitable for applications related to business process modelling.

We have introduced the notions of obligation, permission, prohibition and violation and we have shown how to represent them in a rule based formalism. A rule based formalism gives a representation close to the representation of norms in a normative system. Essentially, every norm is mapped to a rule.

A close and faithful representation is the first step for successful modelling. The second step is reasoning. In the framework presented in this chapter reasoning is done in two phases. In the first phase we use a normalisation procedure to generate all and only (maximal) reparation chains (norms) corresponding to all implicit norms that can be obtained from a given, explicit set of norms: the norms a process has to comply with given in a normative document. The output of the normalisation is the compliance schema for a process. In the second reasoning phase, the task is given a activity (or task) in an instance of a process what normative positions apply to the activity? This part is examined in Section 2.5 where we also gave the rationale for the inference mechanism behind FCL.

3 Process Modelling

We use BPMN notation[6] as our target process description language for a number of reasons. Firstly, graphical business process modelling have a growing acceptance in indus-

[6]`http://www.bpmn.org`

try, and BPMN is steadily increasing its span of adoption. Secondly, BPMN models are conducive to translation to executable process models (e.g., BPEL). Finally, BPMN provides a suitable environment for supporting interactions defined by business contracts because it allows for support of process descriptions that range from internal processes to complex cross-organisational processes, involving several parties. Namely, using BPMN it is possible to describe abstract (or public) processes between parties, where the focus is on exchange of messages between them. In this case, it is possible to either abstract away the internal processes of both parties or to abstract away the internal processes of the other partner only, depending on the circumstances. In the last case, the aim is to provide description of interactions from the point of view of one party. In the most detailed form, BPMN allows for the description of internal processes for all parties, along with the messages between them. In BPMN terms this is called a collaboration (global) process.

3.1 Execution Semantics

The basic execution semantics of the control flow aspect of a business process model is defined using token-passing mechanisms, as in Petri Nets. The definitions used here extend the execution semantics for process models given by (Vanhatalo et al., 2007) with semantic annotations in the form of effects and their meaning.

A process model is seen as a graph with nodes of various types – a single start and end node, task nodes, XOR split/join nodes, and parallel split/join nodes – and directed edges (expressing sequentiality in execution). The number of incoming (outgoing) edges are restricted as follows: start node 0 (1), end node 1 (0), task node 1 (1), split node 1 (>1), and join node >1 (1). The location of all tokens, referred to as a *marking*, manifests the state of a process execution. An execution of the process starts with a token on the outgoing edge of the start node and no other tokens in the process, and ends with one token on the incoming edge of the end node and no tokens elsewhere (cf. *soundness*, e.g., Wynn et al. (2007)). Task nodes are executed when a token on the incoming link is consumed and a token on the outgoing link is produced. The execution of a XOR (parallel) split node consumes the token on its incoming edge and produces a token on one (all) of its outgoing edges, whereas a XOR (parallel) join node consumes a token on one (all) of its incoming edges and produces a token on its outgoing edge.

3.2 Annotation of Processes

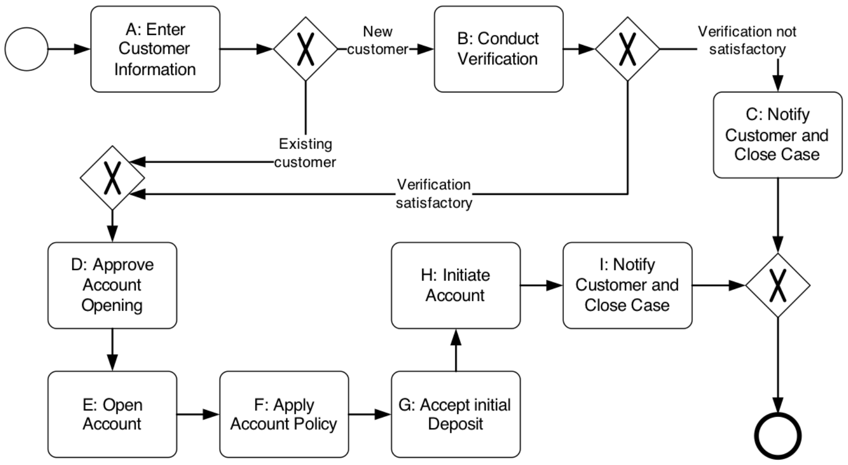
A process model is extended with a set of annotations, where the annotations describe (i) the artefacts or effects of executing a task and (ii) the rules describing the obligations (and other normative positions) relevant for the process.

As for the semantic annotations, the vocabulary is presented as a set of predicates P. There is a set of process variables (x and y in Table 2), over which logical statements can be made, in the form of literals involving these variables. The task nodes can be annotated using *effects* (also referred to as *post-conditions*) which are conjunctions of literals using the process variables. The meaning is that, if executed, a task changes the state of the world according to its effect: every literal mentioned by the effect is true in the resulting world; if a literal l was true before, and is not contradicted by the effect,

Control Objective	**Internal Control**
Customer due diligence	All new customers must be scanned against provided databases for identity checks.
	Accounts must maintain a positive balance, unless approved by bank manager, or for VIP customers.
Record keeping	Retain history of identity checks performed.

Table 1: Control objectives for the process in Figure 2

then it is still true (i.e., the world does not change of its own accord).



A: Enter Customer Information
New customer
B: Conduct Verification
Verification not satisfactory
C: Notify Customer and Close Case
Existing customer
Verification satisfactory
D: Approve Account Opening
H: Initiate Account
I: Notify Customer and Close Case
E: Open Account
F: Apply Account Policy
G: Accept initial Deposit

Figure 2: Example of an account opening process in private banking

The obligations for this example are motivated by the following scenario: A new legislative framework has recently been put in place in Australia for anti-money laundering. The first phase of reforms for the *Anti-Money Laundering and Counter-Terrorism Financing Act 2006* (AML/CTF) covers the financial sector including banks, credit unions, building societies and trustees and extends to casinos, wagering service providers and bullion dealers. The AML/CTF act imposes a number of obligations which include: customer due diligence (identification, verification of identity and ongoing monitoring of transactions), reporting (suspicious matters, threshold transactions and international funds transfer instructions), and record keeping. AML/CTF does not dictate specific conditions but sets out principles businesses have to obey to. Hence businesses need to determine the exact manner in which they will fulfil the obligations, which comprises the design of internal controls specific to the organisation.

Table 1 contains a natural language description of the control objectives and corresponding internal controls for this process; Table 2 shows the semantic effect annotations of the process activities. In this case, the control objectives describe the principles the business is subject to, and the internal controls are the resulting “norms” implementing the control objectives. Control objectives and internal controls are the typical way in which enterprises implement norms regulating they business, processes and procedure.

The control objectives in Table 1 can be expressed by the following FCL rules to create the compliance rule base:

Task	Semantic Annotation
A	$newCustomer(x)$
B	$checkIdentity(x)$
C	$checkIdentity(x)$, $recordIdentity(x)$
E	$owner(x,y)$, $account(y)$
F	$accountType(y,type)$
G	$positiveBalance(y)$
H	$\neg positiveBalance(y)$
I	$accountActive(y)$
J	$notify(x,y)$

Table 2: Annotations for the process in Fig 2.

- All new customers must be scanned against provided databases for identity checks.

  $$r_1 : newCustomer(x) \Rightarrow OcheckIdentity(x)$$

  The meaning of the predicate *newCustomer*(x) is that the input data with $Id = x$ is a new customer, for which we have the obligation to check the provided data against provided databases *checkIdentity*(x). The obligation resulting from this rule is a non-persistent obligation, i.e. as soon as a check has been performed, the obligation is no longer in force.

- Retain history of identity checks performed.

  $$r_2 : checkIdentity(x) \Rightarrow OrecordIdentity(x)$$

  This rule establishes that there is a permanent obligation to keep record of the identity corresponding to the (new) customer identified by x. In addition this obligation is not fulfilled by the achievement of the activity (for example, by storing it in a database). We have a violation of the condition, if for example, the record x is deleted from the database.

- Accounts must maintain a positive balance, unless approved by a bank manager, or for VIP customers.

  $$r_3 : account(x) \Rightarrow OpositiveBalance(x) \otimes OapproveManager(x)$$

  The primary obligation is that each account has to maintain a positive balance *positiveBalance*; if this condition is violated (for any reason the account is not positive), then we still are in an acceptable situation if a bank manager approves the account not to be positive. In this case, the obligation of approving persists until a manager approves the situation; after the approval, the obligation is no longer in force.

  $$r_4 : account(x), owner(x,y), accountType(x,VIP) \Rightarrow P\neg positiveBalance(x)$$

  This rule creates an exception to rule r_3. Accounts of type VIP are allowed to have a non positive balance and no approval is required for this type of accounts (this is achieved by imposing that rule r_4 is stronger than rule r_3, $r_4 \prec r_3$). Notice that the normative position associated to r_4 is a permission.

3.3 Summary

In this section we argued about the advantages about the graphical notation for business processes and we have given an overview of the execution semantics of business processes. However, this is not enough for compliance. While graphical notation can be used to check the structural compliance of a process (i.e., whether task are executed in a prescribed order), graphical notation must be supplemented by annotations. For compliance one has to have two different types of annotations: on one hand we need to know how data is affected by the different task, thus we annotate the single node in a process graph (and eventually the arcs) with the effects produced by the nodes. On the other hand, we have to know the rules (control objectives and internal controls) a process is subject to. Thus, we annotate processes with set of FCL rules describing the normative part regulating the process.

4 Compliance Checking

As stated, our aim in the compliance checking is to figure out (a) which obligations will definitely appear when executing the process, and (b) which of those obligations may not be fulfilled.

In a way, FCL constraint expressions for a normative document define a behavioural and state space which can be used to analyse how well different behaviour execution paths of a business process comply with the FCL constraints. Our aim is to use this analysis as a basis for deciding whether execution paths of a business process are compliant with the FCL and thus with the normative document modelled by the FCL specifications. The central part of this compliance checking is given by the notions of ideal, sub-ideal, non-ideal and irrelevant situations which will be introduced and defined after two simple motivating examples are given.

Consider the following FCL obligation rule:

$$service_1 : WeekDay, FaultMessageEvent \Rightarrow O_S Repair24hours$$

stating that on a week day, when a fault message occurs, the service provider is obliged to repair the fault within 24hrs.

Assume now that one possible execution path from a process is:

1. a *FaultMessageEvent* is received from a premium customer on a week day
2. the service provider reacts by (in the order):
   (a) sending an apology message,
   (b) repairing the fault within 24 hours and
   (c) sending a reparation confirmation message

When checking compliance of this execution path with the obligation it is obvious that the obligation is fulfilled because the fault is fixed within 24 hours. Notice that the execution path also includes additional conditions such as sending of two additional messages (an apology message, and a reparation confirmation message) which are not critical for the obligation.

Consider another example:

$$service_2 : WeekDay, PremiumCustomer, FaultMessageEvent \Rightarrow O_S Repair12hours$$

This reflects the requirement for a faster reaction time for premium customers. Assume we have the following situation:

$$WeekDay, FaultMessageEvent$$

Obviously, this situation is not sufficient for the $O_S repair12hours$ to be activated.

4.1 Ideal Semantics

We now introduce the concepts of ideal, sub-ideal and non-ideal situations to describe various degrees of compliance between execution paths and FCL constraints. We will also provide a semantic interpretation of FCL rules in terms of ideal, sub-ideal, non-ideal and irrelevant situations, which we refer to as Ideal Semantics. In this context, a situation is the state of a process after the execution of a task. Thus, a situation corresponds to the set of effects (literals) obtained after the execution of a task.

Intuitively, an *ideal* situation is a situation where execution paths do not violate FCL expressions, and thus the execution paths (which will then correspond to processes that are related to the contract) are fully compliant with the normative specifications. A *sub-ideal* situation is a situation where there are some violations, but the norms relevant for the situation at hand establish means to recover from the violation, and the compensatory measures have been taken. In other terms there is at least a reparation chains where the primary obligation has not been fulfilled but some of the secondary obligation have been fulfilled. Accordingly, processes resulting in sub-ideal situations are still compliant to a normative document even if they provide non-optimal performances of the normative specifications. A situation is *non-ideal* if it violates a normative document (and the violations are not repaired). In this case, a process resulting in a non-ideal situation does not comply with the normative specifications. There are two possible reasons for a process not to comply with the normative specifications: 1) the process executes some tasks which are prohibited by the normative specifications (or equivalently, it executes the opposite of obligatory tasks); 2) the process fails to execute some tasks required by the normative specifications. Finally, a situation is irrelevant for a normative document if no rule is applicable in the situation. Irrelevant situations correspond to states of affairs where a normative document is silent about them.

As discussed in Section 2.4, for every FCL representation of a contract its normal form contains all conditions that can be derived from the normative specifications and redundant clauses are removed. Thus, normal forms are the most appropriate means to determine whether a process conforms with a normative document. We now define conditions under which we are able to determine whether a situation complies with a set of normative specifications or if it represents a violation of some clauses.

First of all, we define when a situation (set of literals) is either ideal, sub-ideal, non-ideal or irrelevant with respect to a contract rule.

Definition 2

- *A situation S is* ideal *with respect to a rule $\Gamma \Rightarrow A_1 \otimes \cdots \otimes A_n$ iff $\Gamma \cup \{A_1\} \subseteq S$.*

- *A situation S is* sub-ideal *with respect to a rule $\Gamma \Rightarrow A_1 \otimes \cdots \otimes A_n$ iff $\Gamma \cup \{A_i\} \subseteq S$, for some $1 < i \leq n$ such that $\forall A_j$, $j < i$, $A_1, \ldots, A_j \notin S$.*

- *A situation S is* non-ideal *with respect to a rule $\Gamma \Rightarrow A_1 \otimes \cdots \otimes A_n$ iff $\Gamma \subseteq S$ and S is neither ideal nor sub-ideal, i.e., $A_1, \ldots, A_n \notin S$.*

- *A situation S is* irrelevant *with respect to a rule $\Gamma \Rightarrow A_1 \otimes \cdots \otimes A_n$ iff it is neither ideal nor sub-ideal nor non-ideal, i.e., $\Gamma \not\subseteq S$*

Returning to our first example of the previous section, rule *service*$_1$, we have that

$$\Gamma = \{\mathit{WeekDay}, \mathit{FaultMessageEvent}\}$$

Accordingly, for the situation:

$$S = \{\mathit{WeekDay}, \mathit{FaultMessageEvent}, \mathit{SendApologyMessage}, \\ \mathit{Repair24hours}, \mathit{SendReparationConfirmationMessage}\}$$

we have that $\Gamma \cup \{\mathit{Repair24hours}\} \subseteq S$, thus the situation is classified as ideal. As we have seen in the previous section, for the second rule, i.e., *service*$_2$, Γ is not a subset of S, thus S is irrelevant for the rules concerning premium customers.

According to Definition 2, a situation is ideal with respect to a norm if the rule is not violated; sub-ideal when the primary obligation is violated but the rule allows for a reparation, which is satisfied; non-ideal when the primary obligation and all its reparations are violated, and irrelevant when the rule is not applicable. Definition 2 is concerned with the status of a situation with respect to a single rule, while a contract consists of many rules, thus we have to extend this definition to cover the case of a set of rules. In particular we will extend it considering all rules in the normal form for a normative document containing all rules inherent to the document.

Definition 3

- *A situation S is* ideal *with respect to a set of rules R iff for every rule in R, either S is irrelevant or ideal for the rule.*

- *A situation S is* sub-ideal *with respect to a set of rules R iff there is a rule in R for which S is sub-ideal, and there is no rule in R for which S is non-ideal.*

- *A situation S is* non-ideal *with respect to a set of rules R iff there is a rule in R for which S is non-ideal.*

- *A situation S is* irrelevant *with respect to a set of rules R iff for all rules in R the situation S is irrelevant.*

Definition 3 follows immediately from the intuitive interpretation of ideality and the related notions we have provided in Definition 2. On the other hand, the relation between a normal form and the normative specifications from which it is obtained seems to be a more delicate matter. A careful analysis of the conditions for constructing an FCL normal form allows us to state the following general criterion:

Definition 4 *A situation S is ideal (sub-ideal, non-ideal, irrelevant) with respect to a set of FCL rules if S is ideal (sub-ideal, non-ideal, irrelevant) with respect to the normal form of the set of FCL rules.*

It is worth noting that Definition 4 shows the relevance of the distinction between a set of normative specifications and its normal form. This holds in particular for the case of sub-ideal situations. Suppose you have the following set of FCL rules

$$\Rightarrow OA \qquad \neg A \Rightarrow OB$$

The corresponding normal form is

$$\Rightarrow OA \otimes OB$$

While the situation with $\neg A$ and B is sub-ideal with respect to the latter, it would be non-ideal for the former. In the first case, even if $\neg A \Rightarrow OB$ expresses in fact an implicit reparational obligation of the rule $\Rightarrow A$, this is not made explicit. The key point here is that there was no link between the primary and reparation obligations in the original set of rules, but this is made explicit in the normal form. So, there exists a situation which apparently accomplishes a rule and violates the other without satisfying any reparation. This conclusion cannot be accepted because it is in contrast with our intuition according to which the presence of two rules like $\Rightarrow OA$ and $\neg A \Rightarrow OB$ must lead to a unique regulation. For this reason, we can evaluate a situation as sub-ideal with respect to a set of FCL rules only if it is sub-ideal with respect to its normal form.

4.2 Checking Compliance

The ideal semantics allows us to relate business processes and FCL expression, thus it enable us to determine whether a business process is compliant with a set of regulations. What we have to do now is to look at the details of how to relate the two domains. The idea is as follows:

1. We traverse the graph describing the business process and we identify the sets of effects (sets of literals) for all the tasks (nodes) in the process according to the execution semantics specified in Section 3.1.

2. For each task we use the set of effects for that particular task to determine the normative positions (obligations, permissions, prohibitions) triggered by the execution of the task. This means that effects of a task are used as a set of facts, and we compute the conclusions of the defeasible theory resulting from the effects and the FCL rules annotating the process (see Section 2.5). In the same way we accumulate effects we accumulate (undischarged) obligations from one task in the process to the task following it in the process.

3. For each task we compare the effects of the tasks and the obligations accumulated up to the task. If an obligation is fulfilled by a task, we discharge the obligation, otherwise if the obligation is violated we signal the violation. Finally, if an obligation is not fulfilled nor violated, we keep the obligation in the stack of obligations and we propagate the obligation to the successive tasks.

Here, we assume that the obligations derived from a task should be fulfilled in the remaining of the process. Variations of this schema are possible. For example, one could stipulate that the obligations derived from a task should be fulfilled by the tasks immediately after the task[7]. In another approach one could use a schema where for each task one has both preconditions and effects. Then the obligations derived from the preconditions must be fulfilled by current task (i.e., the obligations must be fulfilled by the effects of the task), and the obligations derived from the effects are as in our basic schema.

In the rest of the section we discuss in details step 2 (Section 4.3) and 3 (Section 4.4) above.

4.3 From Tasks to Obligations

The second step to perform when we have to determine whether a process is compliant is to determine the obligations derived by the effects of a task. Given a set of rules R and a set of literals S (plain literals and deontic literals), we can use the inference mechanism of defeasible logic (Section 2.5) to compute the set of conclusions (obligations) in force given the set of literals. These are the obligations an agent has to obey to in the situation described by the set of literals. However, the situation could already be sub-ideal, i.e., some of the obligations prescribed by the rules are already violated. Thus, given a set of literals describing a state-of-affairs one has to compute not only the current obligations, but also what reparation chains are in force given the set.

For example consider a scenario where we have the rules $A \Rightarrow OB$ and $\neg B \Rightarrow OC$, and the effects are A and $\neg B$. The normal form of the rules is $A \Rightarrow OB \otimes OC$ and $\neg B \Rightarrow OC$. The only obligation in force for this scenario is OC. Since we have a violation of the first rule ($A \Rightarrow OB$ and $\neg B$), then we know that it is not possible to have an ideal situation here. Hence, computing only the current obligation does not tell us the state of the corresponding business process. What we have to do is to identify the chain for the ideal situation for the task at hand. To deal with this issue we have to identify the *active* reparation chains.

A reparation chain C is *active* given a set of literals S, if

- there is a rule $\Gamma \Rightarrow C$ such that $\Gamma \subseteq S$, i.e., the rule is triggered by the situation, and
- for all rule for conflicting chains[8], either
  - the chain is not triggered by the situation or
  - the negation of an element before the conflicting element is not in the situation.

Let us examine the following example. Consider the rules

$$\begin{aligned}
&r_1 : A_1 \Rightarrow OB \otimes OC,\\
&r_2 : A_2 \Rightarrow O\neg B \otimes OD,\\
&r_3 : A_3 \Rightarrow OE \otimes O\neg B.
\end{aligned}$$

[7] This approach can be used to check the compliance of the flow of tasks.

[8] Given a chain $A_1, \ldots, A_n$, a conflicting chain is any containing $\neg A_i$, for some $1 \leq i \leq n$.

The effects describing the situation are A_1 and A_3. In this scenario the active chains are $OB \otimes OC$ and $OE \otimes O\neg B$. The chain $OB \otimes OC$ is active since r_2 cannot be used to activate the chain $O\neg B \otimes OD$. For r_3, and the resulting chain $OE \otimes O\neg B$, we do not have the violation of the primary obligation OE of the rule (i.e., $\neg E$ is not one of the effects), so the resulting obligation $O\neg B$ is not entailed by rule r_3.

Consider, again, the scenario described at the end of Section 2.5. Given the rule

$$r : \mathit{Invoice} \Rightarrow O_P \mathit{PayWithin7Days} \otimes O_P \mathit{PayWithInterest}$$

and the case data *Invoice* we have that the active chain is

$$O_P \mathit{PayWithin7Days} \otimes O_P \mathit{PayWithInterest},$$

and the current obligation in force is $O_P \mathit{PayWithin7Days}$. In case of a late payment of the invoice, i.e., when the case data consists of both *Invoice* and $\neg \mathit{PayWithin7Days}$, the active chain is still the same, but the obligation in force is $O_P \mathit{PayWithInterest}$. In the first case, to obey the current obligation (i.e., to pay the invoice in time) results in an ideal situation. In the second case, the best one can do is to end up in a sub-ideal situation.

4.4 Obligation Propagation

A reparation chain is in force if there is a rule of which the reparation chain is the consequent and a set of facts (effects of a task in a process) including the rule antecedents. In addition, we assume that, once in force, a reparation chain remains as such unless we can determine that it has been violated or the obligations corresponding to it have all been obeyed to (these are two cases when we can discharge an obligation or reparation chain). This means that it is not possible to have two instances at the same time of the same reparation chain. Accordingly, a reparation chain in force is uniquely determined by the combination of the task T when the chain has been derived and the rule R from which the chain has been obtained.

The procedure for compliance checking is based on two algorithms, *ComputeObligations* and *CheckCompliance*. *ComputeObligations* is the algorithm to determine the active chains presented in Section 4.3. Given a set of literals S, corresponding to effects of a task T in a process model, we use the algorithm *ComputeObligations* to determine the current set of obligations for the process *Current*. The set of the current obligations includes the new obligations triggered by the task, as well as the obligations carried out from previous tasks. The algorithm *CheckCompliance* scans all elements of *Current* against the set of literals S, and determines the state of each reparation chain ($C = A_1 \otimes A_2$) in *Current*. *CheckCompliance* operates as follows:

```
if $A_1 = OB$, then
  if $B \in S$, then
    remove($[T, R, A_1 \otimes A_2]$, $\mathit{Current}$)
    remove($[T, R, A_1 \otimes A_2]$, $\mathit{Unfulfilled}$)
    if $[T, R, B_1 \otimes B_2 \otimes A_1 \otimes A_2] \in \mathit{Violated}$ then
      add($[T, R, B_1 \otimes B_2 \otimes A_1 \otimes A_2]$, $\mathit{Compensated}$)
  if $\neg B \in S$, then
```

```
    add($[T,R,A_1 \otimes A_2,B]$, $Violated$)
    add($[T,R,A_2]$, $Current$)
  else
    add($[T,R,A_1 \otimes A_2]$, $Unfulfilled$).
```

Let us examine the *CheckCompliance* algorithm. Remember the algorithm scans all active reparation chains one by one. Then for each of them reports on the status of it. For each chain in *Current* (the set of all active chains), it looks for the first element of the chain and it determines the content of the obligation (so if the first element is OB, the content of the obligation is B). Then it checks whether the obligation has been fulfilled (B is in the set of effects), or violated ($\neg B$ is in the set of effects), or simply we cannot say anything about it (none of B and $\neg B$ is in the set of effects). In the first case we can discharge the obligation and we remove the chain from the set of active chains (similarly if the obligation was carried over from a previous task, i.e., it was in the set *Unfulfilled*). In case of a violation, we add the information about it in the system. This is done by inserting a tuple with the identifier of the chain and what violation we have in the set *Violated*. In addition, we know that violations can be compensated, thus if the chain has a second element we remove the violated element from the chain and put the rest of the chain in the set of active chains. Here we take the stance that a violation does not discharge an obligation, thus we do not remove the chain from the set of active chains[9]. Finally, in the last case, the set of effects does not tell us if the obligation has been fulfilled or violated, so we propagate the obligation to the successive tasks by putting the chain in the set *Unfulfilled*. The algorithm also checks whether a chain/obligation was previously violated but it was then compensated.

Definition 5

- *A process is* compliant *iff for all* $[T,R,A] \in Current$, $A = OB \otimes C$, *for every* $[T,R,A,B] \in Violated$, $[T,R,A,B] \in Compensated$ *and* $Unfulfilled = \emptyset$.
- *A process is* fully compliant *iff for all* $[T,R,A] \in Current$, $A = OB \otimes C$, $Violated = \emptyset$ *and* $Unfulfilled = \emptyset$.

The above definition relates the state of a process base on the report generated by the *CheckCompliance* algorithm and the ideal semantics for FCL expressions. In particular, a process is compliant if the situation at the end of the process is at least sub-ideal (it is possible to have violations but these have been compensated for). Similarly a process is fully compliant if it results in an ideal situation.

According to Definition 5, a process is not compliant if the set of unfulfilled obligations (*Unfulfilled*) is not empty. Consider, for example the rule

$$r_3 : account(x) \Rightarrow OpositiveBalance(x) \otimes OapproveManager(x)$$

[9]Governatori et al. (2007) propose a more fine grained classification of obligations. Accordingly it is possible to have obligations that are discharged when are violated, as well as obligations that persist in case of a violation. The above algorithm can be easily modified to deal with the different types of obligations examined by Governatori et al. (2007).

relative to the process of Figure 2 with the annotation as in Table 2. After task E we have, among others, the effect $account(x)$. This means that after task E we have the chain

$$[E, r_4, OpositiveBalance(x) \otimes OapproveManager(x)]$$

in *Current* for task F. After task F, the above entry for the chain obtained from rule r_4 is moved to the set *Unfulfilled*. Suppose now that tasks G and H do not have any annotation attached to them. In this case at the end of the process we still have the active chain, but the resulting situation is not ideal: the antecedent of the rule is a subset of the set of effects, but we do not have the first element of the chain as one of the effects. Thus something the process were required to do was not done; hence, the process is not compliant.[10]

4.5 Summary

In this section we have first introduced a semantics to evaluate a set of literals given a set of FCL expressions determining whether obligations relative to a situation (state-of-affairs) where met or not. Accordingly, a state can be ideal, if all (primary) obligations are fulfilled, sub-ideal, if some obligations are not fulfilled but they are repaired, and non-ideal if there are violations which are not compensated.

Then, we discussed a mechanism to determine whether a process is compliant or not based on the ideal semantics. In particular, first we have to gather the effects for a task (and propagate effects from one task to tasks after it), then we have to determine the active obligations as well as the situation is ideal, sub-ideal or non-ideal with respect to the effects of the current task. However, a task could introduce some degree of non-compliance that could be resolved in successive tasks. We have show how to propagate obligations and the compliance status across task. We also have discussed some variants of the main schema.

5 Related Work

Governance, risk and compliance (GRC) is an emerging area of research which holds challenges for various communities including information systems, business software development, legal, cultural, behavioural studies and corporate governance.

In this chapter, we have focused on compliance management from an information systems perspective, in particular the modelling and analysis of compliance requirements. Both process modelling as well as modelling of normative requirements are well studied fields independently, but until recently the interactions between the two have

[10]What about a situation where after task F we have a task producing the annotation *approveManager*(x) but no task with effect *positiveBalance*(x)? Is the resulting process compliant? In this case we the reparation of the violation, but not the violation. The issue here is that we could have that a sanction is enforced before the violation the sanction was supposed to compensate occurred. Thus we are in a situation similar to that described in footnote 9 where the way to address the issue depends on the types of the obligations we have to deal with. Anyway, (i) it is easy to modify algorithm *CheckCompliance* to account for this type of cases, (ii) if one accepts pre-emptive reparations one can change the definition that classifies a process as compliant by replacing the condition that $Unfulfilled = \emptyset$ with the condition: let S be the set of effects for the end task of a process, $\forall [T, R, OA_1 \otimes \cdots \otimes OA_n] \in Unfulfilled$, $\exists A_i \in S$.

been largely ignored (Desai et al., 2005; Padmanabhan et al., 2006). In particular, zur Muehlen et al. (2007) provide a valuable representational analysis to understand the synergies between process modelling and rule modelling.

It is obvious that the modelling of controls will be undertaken as rules, although the question of appropriate formalism is still under studied. A plethora of proposals exist both in the research community on formal modelling of rules, as well as in the commercial arena through business rule management systems.

Historically, formal modelling of normative systems has focused on how to capture the logical properties of the notions of the normative concepts (e.g., obligations, prohibitions, permissions, violations, . . .) and how these relate to the entities in an organisation and to the activities to be performed. Deontic logic is the branch of logic that studies normative concepts such as obligations, permissions, prohibitions and related notions. Standard Deontic Logic (SDL) is a starting point for logical investigation of the basic normative notions and it offers a very idealised and abstract conceptual representation of these notions but at the same time it suffers from several drawbacks given its high level of abstraction (Sartor, 2005). Over the years many different deontic logics have been proposed to capture the different intuitions behind these normative notions and to overcome drawbacks and limitations of SDL. One of the main limitations in this context is its inability to reason with violations, and the obligations arising in response to violations (Carmo and Jones, 2002). Very often normative statements pertinent to business processes, and in particular contracts, specify conditions about when other conditions in the document have nor been fulfilled, that is when some (contractual) clauses have been violated. Hence, any formal representation, to be conceptually faithful, has to been able to deal with this kind of situations.

As we have discussed before compliance is a relationship between two sets of specifications: the normative specifications that prescribe what a business has to do, and the process modelling specification describing how a business performs its activities. Accordingly to properly verify that a process/procedure complies with the norms regulating the particular business one has to provide conceptually sound representations of the process on one side and the norms on the other, and then check the alignment of the formal specifications of the process and the formal specifications for the norms.

In this chapter, we have proposed a formal modelling of controls through the Formal Contract Language (FCL). FCL has proved effective due to its ability to express reparation chains and consequently ability to reason with violations.

There have been some other notable contributions from research on the matter of control modelling. Goedertier and Vanthienen (2006) presents a logical language PENELOPE, that provides the ability to verify temporal constraints arising from compliance requirements on effected business processes. Küster et al. (2007) provide a method to check compliance between object life-cycles that provide reference models for data artefacts e.g. insurance claims and business process models. Giblin et al. (2006) provide temporal rule patterns for regulatory policies, although the objective of this work is to facilitate event monitoring rather than the usage of the patterns for support of design time activities. Furthermore, Agrawal et al. (2006) presented a workflow architecture for supporting Sarbanes-Oxley Internal Controls, which include functions such as workflow modelling, active enforcement, workflow auditing, as well as anomaly detection.

Another line of investigation studies compliance based on the structure of business processes. Ghose and Koliadis (2007) consider an approach where the tasks of a business process model, written in BPMN, are annotated with the effects of the tasks, and a technique to propagate and cumulate the effects from a task to a successive contiguous one is proposed. The technique is designed to take into account possible conflicts between the effects of tasks and to determine the degree of compliance of a BPMN specification. Chopra and Sing (2007), on the other hand, investigate compliance in the context of agents and multi-agent systems based on a classification of paths of tasks. Roman and Kifer (2007) proposed Concurrent Transaction Logic to model the states of a workflow and presented some algorithms to determine whether the workflow is compliant. The major limitation of these approaches to compliance is that they ignore the normative aspects of compliance.

There has been some complementary work in the analysis of formal models representing normative notions. For example, Farrell et al. (2005) study the performance of business contract based on their formal representation. Desai et al. (2008) seek to provide support for assessing the correctness of business contracts represented formally through a set of commitments. The reasoning is based on value of various states of commitment as perceived by cooperative agents. Research on closely related issues has also been carried out in the field of autonomous agents (Alberti et al., 2006).

As discussed previously, modelling the controls is only the first step towards compliance by design. The second essential step is the enrichment of process models with compliance requirements (i.e., the modelled controls). Clearly this cannot take place without a formal controls model (as proposed by above mentioned works), or at least some machine readable specification of the controls. There have been recently some efforts towards support for business process modelling against compliance requirements. In particular, the work of zur Muehlen and Rosemann (2005) provides an appealing method for integrating risks in business processes. The proposed technique for "risk-aware" business process models is developed for EPCs (Event-Driven Process Chains) using an extended notation. Sadiq et al. (2007) propose an approach based on control tags to visualise internal controls on process models. Liu et al. (2007) takes a similar approach of annotating and checking process models against compliance rules, although the visual rule language, namely BPSL is general purpose and does not directly address the deontic notions providing compliance requirements.

Lastly, although this section has primarily focused on preventative approaches to compliance, it is important to identify the role of detective approaches as well, where a wide range of supporting technologies are present. These include several commercial solutions such as business activity monitoring, business intelligence etc. Noteworthy in research literature with respect to compliance monitoring, is the synergy with process mining techniques (van der Aalst et al., 2003) which provide the capability to discover runtime process behaviour (and deviations) and can thereby assist in detection of compliance violations.

6 Conclusions

The growing importance of governance, risk and compliance for various industries, has created an evident need to provide supporting tools and methods to enable organisations seeking compliance, which may ranging from safeguards against enforceable undertakings to being champions of corporate social responsibility. The challenges that reside in this topic warrant systematic approaches that motivate and empower business users to achieve a high degree of compliance with regulations, standards, and corporate policies.

Process and control modelling represent two distinct but mutually dependent specifications in current enterprise systems. In this chapter, we take the view that the two specifications, will be created somewhat independently, at different times, and by different stakeholders, using their respective conceptually faithful representation schemes. However the convergence of the two must be supported in order to achieve business practices that our compliant with control objectives stemming from various regulatory, standard and contractual concerns. This convergence should be supported with a systematic and well structured approach if the vision of compliance by design is to be achieved.

We have proposed a means of achieving so called compliance by design through an overall methodology that can be summarised into three main steps of control modelling; process enrichment; and compliance checking through analysis and feedback for compliance aware process (re)design. Figure 3 summarises the overall methodology that provides a structured and systematic approach to undertaking changes in the process model in response to compliance requirements.

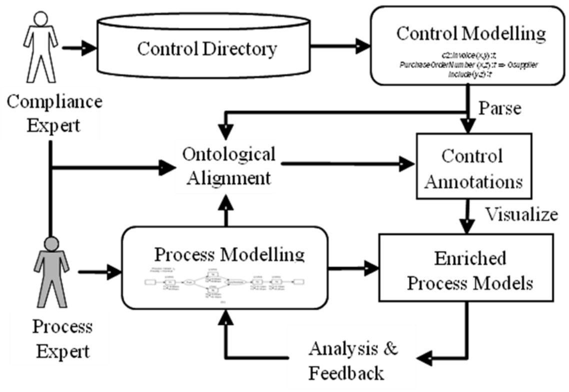


Figure 3: Summary of Overall Methodology

The proposed language for control modelling, namely FCL (section 2), provides a conceptually faithful representation of the compliance requirements. In addition FCL offers two reasoning modules: (1) a normaliser to make explicit rules that can be derived from explicitly given rules by merging their normative conclusions, to remove redundancy and identify conflicts rules; and (2) an inference engine to derive conclusions given some propositions as input (Governatori, 2005). The rigour introduced by FCL enables a systematic establishment of control models with process models. As outlined in section 3, process enrichment can be realised as a result through structured control annotations. These annotations can not only provide a means of visualising the impact of compliance controls on process models (Sadiq et al., 2007), but also assist in compliance checking (section 4) and analysis and feedback for subsequent (re)design of the process models.

One of the biggest challenges facing the compliance industry is the measurement of adequacy of controls (KPMG Advisory, 2005). The methodology proposed provides the added benefit of providing the capability for diagnostics. That is provide a means of understanding what needs to be done in order to achieve (an acceptable degree of) compliance (Lu et al., 2007). This has the potential to create a more holistic approach to compliance management, by not only providing preventative and detective techniques, but also corrective recommendations. This allows organisations to better respond to the changing regulatory demands and also reap the benefits of process improvement. We recommend that future research endeavours in this area should strive towards compliance management frameworks that provide a close integration of the process and control modelling perspectives.

Acknowledgments

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

References

Agrawal, R., C. M. Johnson, J. Kiernan, and F. Leymann (2006). Taming compliance with sarbanes-oxley internal controls using database technology. In L. Liu, A. Reuter, K.-Y. Whang, and J. Zhang (Eds.), *ICDE*, pp. 92. IEEE Computer Society.

Alberti, M., M. Gavanelli, E. Lamma, F. Chesani, P. Mello, and P. Torroni (2006). Compliance verification of agent interaction: a logic-based software tool. *Applied Artificial Intelligence 20*(2-4), 133–157.

Alonso, G., P. Dadam, and M. Rosemann (Eds.) (2007). *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings*, Volume 4714 of *Lecture Notes in Computer Science*. Springer.

Antoniou, G., D. Billington, G. Governatori, and M. J. Maher (2001). Representation results for defeasible logic. *ACM Transactions on Computational Logic 2*(2), 255–287.

Antoniou, G., D. Billington, G. Governatori, and M. J. Maher (2006). Embedding defeasible logic into logic programming. *Theory and Practice of Logic Programming 6*(6), 703–735.

Carmo, J. and A. J. Jones (2002). Deontic logic and contrary to duties. In D. Gabbay and F. Guenther (Eds.), *Handbook of Philosophical Logic, 2nd Edition*, Volume 8, pp. 265–343. Dordrecht: Kluwer.

Chopra, A. K. and M. P. Sing (2007, May 8, 2007). Producing compliant interactions: Conformance, coverage and interoperability. In M. Baldoni and U. Endriss (Eds.), *Declarative Agent Languages and Technologies IV*, Volume 4327 of *LNAI*, Berlin Heidelberg, pp. 1–15. Springer-Verlag.

Desai, N., A. U. Mallya, A. K. Chopra, and M. P. Singh (2005). Interaction protocols as design abstractions for business processes. *IEEE Trans. Software Eng. 31*(12), 1015–1027.

Desai, N., N. C. Narendra, and M. P. Singh (2008). Checking correctness of business contracts via commitments. In L. Padgham, D. C. Parkes, J. Müller, and S. Parsons (Eds.), *AAMAS*, pp. 787–794. IFAAMAS.

Farrell, A. D. H., M. J. Sergot, M. Sallé, and C. Bartolini (2005). Using the event calculus for tracking the normative state of contracts. *International Journal of Cooperative Information Systems 14*(2-3), 99–129.

Ghose, A. and G. Koliadis (2007). Auditing business process compliance. In *Service Oriented Computing, ISOC 2007*, LNCS, pp. 169–180. Springer.

Giblin, C., S. Müller, and B. Pfitzmann (2006, October). From regulatory policies to event monitoring rules: Towards model driven compliance automation. Technical report, IBM Research Report. Zurich Research Laboratory.

Goedertier, S. and J. Vanthienen (2006). Designing compliant business processes with obligations and permissions. In *Business Process Management (BPM) Workshops*, pp. 5–14.

Governatori, G. (2005). Representing business contracts in RuleML. *International Journal of Cooperative Information Systems 14*(2-3), 181–216.

Governatori, G., J. Hulstijn, R. Riveret, and A. Rotolo (2007). Characterising deadlines in temporal modal defeasible logic. In *20th Australian Joint Conference on Artificial Intelligence, AI 2007*, pp. 486–496.

Governatori, G. and A. Rotolo (2006). Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic 4*, 193–215.

KPMG Advisory (2005). *The Compliance Journey: Balancing Risk and Controls with Business Improvement*. KPMG Advisory.

Küster, J. M., K. Ryndina, and H. Gall (2007). Generation of business process models for object life cycle compliance. See Alonso et al. (2007), pp. 165–181.

Liu, Y., S. Müller, and K. Xu (2007). A static compliance-checking framework for business process models. *IBM Systems Journal 46*(2), 335–362.

Lu, R., S. W. Sadiq, and G. Governatori (2007). Compliance aware business process design. In *Business Process Management Workshops*, pp. 120–131.

Nute, D. (1994). Defeasible logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 3.

Padmanabhan, V., G. Governatori, S. Sadiq, R. M. Colomb, and A. Rotolo (2006). Process modelling: The deontic way. In M. Stumptner, S. Hartmann, and Y. Kiyoki (Eds.), *Conceptual Modelling 2006. Proceedings of the Thirds Asia-Pacific Conference on Conceptual Modelling (APCCM2006)*, pp. 75–84. Australian Computer Science Communications.

Pesic, M. and W. M. P. van der Aalst (2006). A declarative approach for flexible business processes management. In J. Eder and S. Dustdar (Eds.), *Business Process Management Workshops*, Volume 4103 of *Lecture Notes in Computer Science*, pp. 169–180. Springer.

Roman, D. and M. Kifer (2007). Reasoning about the behaviour of semantic web services with concurrent transaction logic. In *VLDB*, pp. 627–638.

Sadiq, S., G. Governatori, and K. Naimiri (2007). Modelling of control objectives for business process compliance. See Alonso et al. (2007), pp. 149–164.

Sadiq, S. W., M. E. Orlowska, and W. Sadiq (2005). Specification and validation of process constraints for flexible workflows. *Inf. Syst. 30*(5), 349–378.

Sartor, G. (2005). *Legal Reasoning*. Dordrecht: Springer.

van der Aalst, W. M. P., B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters (2003). Workflow mining: A survey of issues and approaches. *Data Knowl. Eng. 47*(2), 237–267.

Vanhatalo, J., H. Völzer, and F. Leymann (2007). Faster and More Focused Control-Flow Analysis for Business Process Models though SESE Decomposition. In *5th International Conference on Service-Oriented Computing (ICSOC)*, pp. 43–55.

Wynn, M. T., H. Verbeek, W. M. van der Aalst, A. H. ter Hofstede, and D. Edmond (2007). Business process verification - finally a reality! *Business Process Management Journal*.

zur Muehlen, M., M. Indulska, and G. Kemp (2007). Business process and business rule modeling languages for compliance management: A representational analysis. In *Proc ER 2007: Tutorials, Poster, Panels, and Industrial Contribution*.

zur Muehlen, M. and M. Rosemann (2005). Integrating risks in business process models. In *Proceedings of 16th AustralasianConference on Information Systems*.