On compliance of business processes with business contracts

Guido Governatori Zoran Milosevic* Shazia Sadiq and Maria Orlowska
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 072, Australia
{guido,zoran,shazia,maria}@itee.uq.edu.au

February 2, 2007

Abstract

This paper addresses the problem of ensuring compliance of business processes, implemented within and across organisational boundaries, with the constraints stated in related business contracts. In order to deal with the complexity of this problem we propose two solutions that allow for a systematic and increasingly automated support for addressing two specific compliance issues. One solution provides a set of guidelines for progressively transforming contract conditions into business processes that are consistent with contract conditions thus avoiding violation of the rules in contract. Another solution compares rules in business contracts and rules in business processes to check for possible inconsistencies. Both approaches rely on a computer interpretable representation of contract conditions that embodies contract semantics. This semantics is described in terms of a logic based formalism allowing for the description of obligations, prohibitions, permissions and violations conditions in contracts. This semantics was based on an analysis of typical building blocks of many commercial, financial and government contracts. The study proved that our contract formalism provides a good foundation for describing key types of conditions in contracts, and has also given several insights into valuable transformation techniques and formalisms needed to establish better alignment between these two, traditionally separate areas of research and endeavour. The study also revealed a number of new areas of research, some of which we intend to address in near future.

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 [23] or Health Insurance Privacy Act (HIPPA) [24]. 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 [28]. 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 stated in business contracts. The rules in business contracts represent a broad agreement of how two or more parties decided to engage in

*Also with Deontik, Australia, zoran@deontik.com their interactions, but they do not specify local interactions within either of the parties. So, ensuring compliance of business processes with business contracts means ensuring consistency of rules stated in business contracts 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 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.

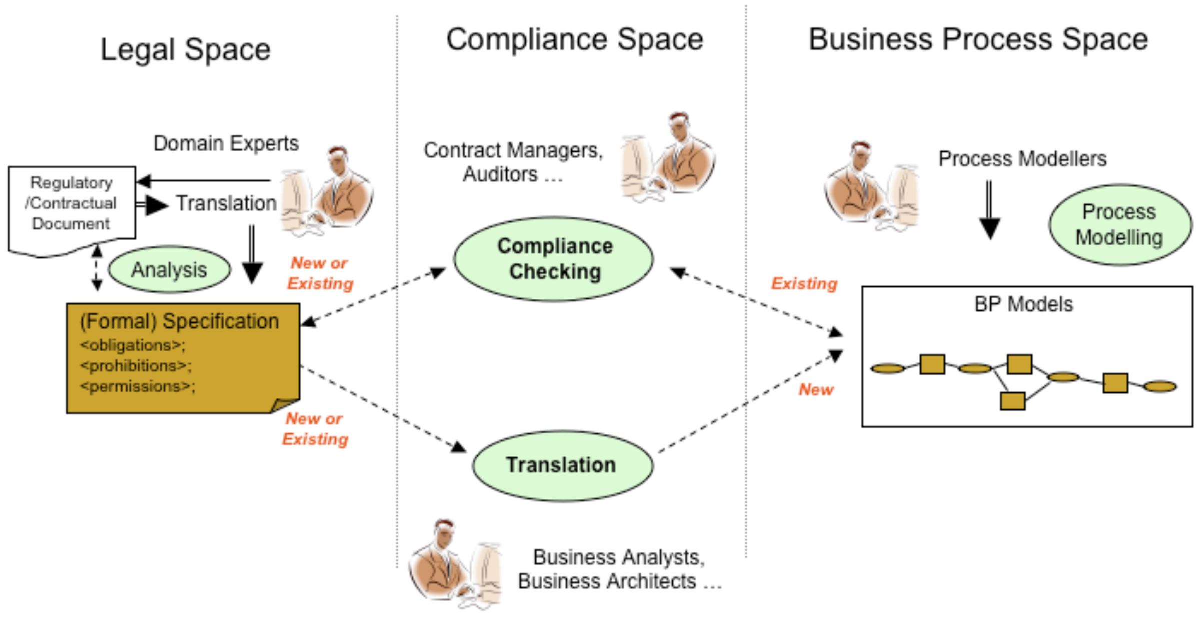

Figure 1: Compliance Space

Compliance of business processes with business contracts 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).

In legal domain, the focus is on contract negotiation, contract drafting and ensuring the legal validity of contracts according to relevant laws such as business contracts law and various regulations. This has been traditionally the domain of legal professionals and in recent times the domain has started involving specialists with certain computing background, needed to support tools-based contract authoring and analysis (see, for example [8]). While most of this analysis has been done through manual checking and analysis of contracts, some recent developments towards formalisation of contract conditions [21], open new possibilities for an increasing role of tools to support contract analysis, e.g., to identify clause inconsistency.

The business process space on the other hand, has been the focus of management science, such as various business process re-engineering 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.

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 with organisations. This new space has thus led to the development of new roles such as compliance auditors, or requirements for new skills to be developed by existing roles, such as of contract managers, business analysts or business architects, for the contract/compliance management domain.

Ensuring compliance of business processes with business contracts is a complex problem, involving a number of activities. In this paper we propose two solution approaches, which when supported by appropriate set of computer tools, would facilitate better compliance between these two different areas.

The first solution approach provides a systematic set of guidelines for progressive refinement of contract conditions into a complaint set of rules for the execution of business processes, covering both cross-organisational and internal business processes (see Figure 1). This approach is of particular value in situations when new business systems are to be designed, permitting for example the design of new business processes to be derived from business contract conditions. This is of further value, when underlying business process management systems are in place or planned to be deployed, ensuring fast deployment of new processes, based on the corresponding business process specifications. It is either new or existing business contracts that can be used as a basis for the design of new business processes, to be contract-complaint. Although the main driver for business process design in process improvement [15], we argue that contract violations can be mitigated when new processes are designed with the explicit understanding of contract conditions and compliance with them.

The second solution approach 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 contracts, opening possibilities for failing to satisfy contractual commitments. This requires checking compliance of processes against business contracts. This can be either against the existing contracts, to fix possible inconsistencies that have not been detected yet, or against new contracts to detect whether existing business processes can lead to conflicts with new contracts, either in terms of incompatible rules or in terms of unrealistic resource expectations that new contracts may require (see Figure 1).

In order to support addressing both of these compliance techniques we needed a formal representation of contracts, which can be used as an intermediate form for both of these approaches. This paper presents our formalism, which was developed based on an analysis of key building blocks of business contracts.

Note that although the paper specifically focuses on contracts as source of policies and constraints, the reasoning will be quite similar when considering the impact of external policies such as regulatory legislature.

The rest of paper is structured as follows. In the next section we consider some typical categories of conditions of contracts, then in Section 3 we introduce a sample contract that will be used for illustrative purposes throughout the paper. In Section 4 we discuss the minimal required features for a formalism for business contracts. Section 5 is dedicated to the presentation of the formalism and its properties and, we show ow the formalism can be used for the representation and analysis of the contract of Section 3. In Section 6 we investigate methodologies to derive compliant processes from normative specifications. In Section 7 we develop an event and state based semantics for our formal language. The aim of this section is to develop a common ground where one can compare formal contract specifications and business process specifications. In Sections 8 we investigate how to check for compliance of business processes with contracts based on the semantics proposed in Section 7.

2 Contracts — key legalese structures

From a system-theoretic point of view, a contract is an agreement that specifies part of the collective behaviour of two or more objects [28]. In the world of law, these objects can be trading partners (individuals or organisations) considered as legal entitles and the agreement reflects their mutual promises, commitments and expectations while being enforceable by law. The behaviour of an object may involve many other aspects of behaviour, either internal behaviour or interactions with other objects which are subject to other agreements and contracts.

From a legal point of view, the contract-governed part of the collective behaviour can typically be expressed through one or more of the legally-centric types of contract conditions. We have identified several categories of such conditions or legalese structures:

1. The declaration of *pre-existing external constraints* from the environment which apply to the contract as a whole or to the variables in the contract. These for example may be rules and policies from various types of law, such as taxation law, employment law or business contracts law;

2. *Definitional expressions* explaining meaning of certain terms in contracts, e.g. that price is nominated in the Australian dollars;

3. The declaration of a *period of validity* when the contract is in effect or a duration for the contract;

4. The statement of *core normative policies* such as obligations, permissions and prohibitions that apply to the parties involved, either directly from the contract in question or from the environment,

as per category 1 above. Note that in the contracts, the core normative policies will typically apply to a subject of a policy but will often mention a target or beneficiary of the policy; in cases where a beneficiary is omitted, it may be inferred from the context;

5. The statement of other enterprise policies that reflect typical terms used by business and which can mostly be reduced on the core policies above; we refer to these as *compound normative policies*; examples are the concepts of rights, liabilities, commitment and responsibility;
6. The statement of policy-related actions that cover transfer of normative modalities between principals and agents, such as various forms of delegation; in this paper we call them *policy-transfer* (delegation) actions;
7. The specification of events that signify occurrence of violations of the policies or the events that signify situations that could potentially lead to some violations in future; in this paper we call them *attention events*;
8. Second-effect policies to be invoked in cases of violations of any of the above policies; we call these *reparation policies*, as per [9];
9. The expression of *force-majeure* conditions, explaining circumstances which are beyond control of either parties; these need to be mentioned, although they might not be able to invoke of any subsequent measures;
10. A number of *structuring constructs* introduced for the purposes of grouping these expressions and supporting reuse of the existing fragments, e.g. a contract clause can consist of a number of other legal statements.

The legalese structures above can be combined in various ways to reflect the specific circumstances that apply to the contract in question.

3 A Sample Contract

This section introduces an example of a business contract, initially introduced in [11]. This is an example of a service contract between an ISP provider and a Purchaser of ISP services.The contract is structured in terms of a number of clause groups, each of which contains contract conditions that will be analysed and formalised in the subsequent sections.

CONTRACT OF SERVICES

This Deed of Agreement is entered into as of the Effective Data identified below.

BETWEEN ABC Company (To be known as the Purchaser)

AND ISP Plus (To be known as the Supplier)
WHEREAS (Purchaser) desires to enter into an agreement to purchase from (Supplier) Application Server (To be known as (Service) in this Agreement).

NOW IT IS HEREBY AGREED that (Supplier) and (Purchaser) shall enter into an agreement subject to the following terms and conditions:

1 Definitions and Interpretations

1.1 Price is a reference to the currency of the Australia unless otherwise stated.

1.2 This agreement is governed by Australia law and the parties hereby agree to submit to the jurisdiction of the Courts of the Queensland with respect to this agreement.

2 Commencement and Completion

2.1 The commencement date is scheduled as January 30, 2006.

2.2 The completion date is scheduled as January 30, 2007.

3 Price Policy

3.1 A “Premium Customer” is a customer who has spent more that $10000 in services. Premium Customers are entitled a 5% discount on new orders.

3.2 Services marked as "special order" are subject to a 5% surcharge. Premium customers are exempt from special order surcharge.

3.3 The 5% discount for premium customers does not apply for services in promotion.

4 Purchase Orders

4.1 The (Purchaser) shall follow the (Supplier) price lists at http://supplier/cat1.html

4.2 The (Purchaser) shall present (Supplier) with a purchase order for the provision of (Services) within 7 days of the commencement date.

5 Service Delivery

5.1 The (Supplier) shall ensure that the (Services) are available to the (Purchaser) under Quality of Service Agreement (http://supplier/qos1.htm). (Services) that do not conform to the Quality of Service Agreement shall be replaced by the (Supplier) within 3 days from the notification by the (Purchaser), otherwise the (Supplier) shall refund the (Purchaser) and pay the (Purchaser) a penalty of $1000.

5.2 The (Supplier) shall on receipt of a purchase order for (Services) make them available within 1 days.

5.3 If for any reason the conditions stated in 4.1 or 4.2 are not met, the (Purchaser) is entitled to charge the (Supplier) the rate of $ 100 for each hour the (Services) are not delivered.

6 Payment

6.1 The payment terms shall be in full upon receipt of invoice. Interest shall be charged at 5 % on accounts not paid within 7 days of the invoice date. The prices shall be as stated in the sales order unless otherwise agreed in writing by the (Supplier).

6.2 Payments are to be sent electronically, and are to be performed under standards and guidelines outlined in PayPal.

7 Termination

7.1 The (Supplier) can terminate the contract after three delayed payments.

8 Disputes NOT SHOWN TO SAVE SPACE

Notice that the above contract contains clauses covering several of the legalese structures presented in Section 2.

4 Contract Formalism

The identification of several categories of legalese contract expression above is a step towards a more structured way of representing contracts. This structuring provides a better starting point for considering the use of several existing logics and/or normative systems theories to arrive at a formal (i.e. logic) representation of contracts. Example of such logics are modal, deontic and temporal logic, along with a recently proposed logic of violations [13]. In many respects the scope of normative systems overlaps with the scope of these logics. This is because they analyse those systems whose behaviour is covered by a set of norms, being statements of constraints on expected behaviour such as what behaviour is allowed, what behaviour must not occur, what behaviour is required and so on.

4.1 From legalese structures to deontic constraints

A detailed examination of the semantics of the contract conditions above reveals that their legal intent and form are closely related to various types of behavioural expressions that apply to the parties in contract. Most of these behavioural expressions are in the form of core normative policies (or deontic modalities), the core of which are obligations, permissions and prohibitions.

Deontic constraints express what parties to the contact are required to perform (obligations), what they are allowed to do (permissions), or what they are not allowed to do (prohibitions).

In general, deontic constraints apply to the behaviour of actors playing roles in some policy context (contract being a special case of a set of policies). This context is typically determined by the way the roles are configured into collaborative structures and by the pre-existing constraints from the environment of this context space. In the specific case of legal contracts, this context is the contract itself, consisting of various types of contract conditions as discussed above, all of which will apply to the two or more roles specified in the contract. These roles in turn could be played by different legal entities, and policies apply to these legal entities when they accept the constraints of the respective roles.

Deontic constraints are structured in terms of:

- *constraints* on behaviour typically expressed in terms of events and relationship between events, closely reflecting the specification of various policies as part of contract conditions. The event (relationship) constraint describes the expected behaviour of the party in question. Events describe the actions of the parties that are subject to the constraint, but also other occurrences such as expiration of deadlines, or actions of other parties.

- the specification of the *modality* that applies to the party, e.g. an obligation, permission or prohibition.

- a *subject* to which modality and behavioural constraints apply. A deontic modality may explicitly identify a *beneficiary* (or target) from the subject side in a modality expression, although the beneficiary may be implied from the contract.

- Triggering conditions which signify that normative policies are in force; these can be temporal events, but also other events, such as violation of other policies.

The semantics of modality can be expressed in terms of a set of observations chosen to determine whether the corresponding policy is satisfied or not. For example, the prohibition modality can be checked through an observation of occurrence(s) which are contrary to the prohibition statement.

Note that the style of constraints in deontic expressions is different from the style of expressions in traditional business process specification languages. The focus of the process languages is on the description of control and data flows to support repeatable and automatable activities. A lesser emphasis is given to the detailed description of various types of associations between roles and process activities, and no emphasis is normally given to the organisational consequences of the violations of agreed behaviour.

Recent business process specifications, however, are increasingly adopting an event-based style of behaviour which better suits needs for more flexibility such as real-time process adaptations. Events can be used to determine process flows in real-time as a result of the outcome of a specific task or to generate notifications of specific events (e.g. natural events or temporal events) or the recognition of an emerging economic or market trend.

We believe that the event-oriented style of expression is suitable for the expression of deontic constraints and the emerging business processes specifications centred on the flow of events can be exploited as a mechanism to facilitate the derivation of business processes compliant with contract conditions.

4.2 From deontic constraints to contract formalism

This section presents a formalism for describing deontic constraints, based on commonly accepted principles of deontic logic, but extended with the formalism for treating violations. The formalism, called the Formal Contract Language (FCL) was first presented in [12], and is based on the logic of violations developed by Governatori and Rotolo [13], used to represent violation in contracts.

4.2.1 Formalising deontic constraints

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

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

The modal 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 contracts: it is possible to execute obligations without doing something that is forbidden.

As stated before, the deontic constraints in contracts apply to the roles involved in the contract, specifically to the subject to which constraints apply, and possibly including the target or beneficiary. 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'. If a beneficiary is mentioned, the expression is extended, i.e. $O_{s,b}A$ to be read as 's has the obligation to do A with respect to b'.

In case of certain breaches of polices in contract by actors playing the roles in a contract, special policies may be included to express the respective obligations for these actors. These policies can vary from pecuniary penalties to the termination of the contract itself. 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. 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 contracts, design subsequent business processes to minimise or deal with such violations and also to support the monitoring of business contracts.

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 [7]. This logic is also suitable to model chains of violations and is described next.

4.2.2 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. This final condition is one which cannot be violated and this it is to be a permission. 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., when $\neg A$ is the case, thus resulting in a violation of the obligation OA), then the obligation OB is activated and becomes in force until it is satisfied or violated. In the latter case a new obligation may be activated, followed by others in chain, as appropriate.

5 Formal Contract Language (FCL)

We now provide a formal account of the idea presented in Section 4.2 which we will refer to as Formal Contract Logic (FCL). FCL was introduced in [11] for the formal analysis of business contracts and it is based on previous work on formal representation of contracts [9], logic of violations [13], and normative positions based on Deontic Logic with Directed Obligations [17, 16]. The language of FCL consists of two sets of atomic symbols: a numerable set of propositional letters $p, q, r, \ldots$, intended to represent the state variables of a contract and a numerable set of event symbols $\alpha, \beta, \gamma, \ldots$ corresponding to the relevant events in a contract; complex events can be obtained from simpler events using the sequence operator ";" (e.g., $\alpha;\beta$ means that event α is followed by event β), conjunction operator $\wedge$ (e.g., $\alpha \wedge \beta$ meaning that both event α and event β are expected to occur), and the disjunction operator $\vee$ (e.g., $\alpha \vee \beta$ meaning that either of the two events α and β are expected to occur). 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 (CTD) operator). The formulas of FCL will be constructed in two steps according to the following formation rules:

- every propositional letter is a literal;
- every event symbol 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 modal literals.

After we have defined the notions of literal and modal 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 modal literal is an $\otimes$-expression;
- if $Ol_1, \ldots, Ol_n$ are modal 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.

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 (represented that A does not hold, i.e., that the negation of A, $\neg A$ 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 policy of a contract is represented by a rule in FCL, where a rule is an expression

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

where r is the name/id of the policy, $A_1, \dots, A_n$, 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 modal 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, the second part of clause 5.1 of the contract ("the supplier shall refund the purchaser 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") can be represented as[1]

$$r : \neg p, \neg\alpha \vdash O_S\beta$$

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

5.1 Normal Forms

We introduce transformations of an FCL representation of a contract to produce a normal form of the same (NFCL). A normal form is a representation of a contract based on an FCL specification containing all contract 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 contract, 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 5.1.1) we describe a mechanism to derive new contract conditions by merging together existing contract clauses. In particular we link an obligation and the obligations triggered in response to violations of the obligation. Then, in Section 5.1.2, we examine the problem of redundancies, and we give a condition to identify and remove redundancies from the formal specification of a contract.

5.1.1 Merging Contract Conditions

One of the features of the logic of violations is to take two rules, or clauses in a contract, 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 policy like (in what follows Γ and Δ are sets of premises)

$$\Gamma \vdash O_sA.$$

Given an obligation like this, if we have that the violation of O_sA is part of the premises of another policy, for example,

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

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

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

[1]In what follows we will use O_S and P_S fot 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.

This reads as follows: given two policies such that one is a conditional obligation ($\Gamma \vdash O_s A$) and the antecedent of second contains the negation of the propositional content of the consequent of the first ($\Delta, \neg A \vdash O_{s'} C$), 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., $\Gamma, \Delta \vdash O_s A \otimes O_{s'} C$) 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 in contracts they are the same.

Suppose the contract includes the rules

$$\begin{gathered} r : \mathit{Invoice} \vdash O_P \mathit{PayWithin7Days} \\ r' : \neg \mathit{PayWithin7Days} \vdash O_P \mathit{PayWithInterest}. \end{gathered}$$

From these we obtain

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

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.

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

For example we can consider the situation described by Clause 5.1 of the contract. Given the rules

$$\begin{gathered} r : \mathit{Invoice} \vdash O_S \mathit{QualityOfService} \otimes O_S \mathit{Replace3days} \\ r' : \neg \mathit{QualityOfService}, \neg \mathit{Replace3days} \vdash O_S \mathit{Refund\&Penalty} \end{gathered}$$

from which we derive the new rule

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

The above patterns are just special instances of the general mechanism described in details in [13, 9].

5.1.2 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 contract. To deal with this issue we introduce the notion of subsumption between rules. Intuitively 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

$$\begin{aligned} r : \mathit{Service} &\vdash O_S \mathit{QualityOfService} \otimes O_S \mathit{Replace3days} \otimes O_S \mathit{Refund\&Penalty}, \\ r' : \mathit{Service} &\vdash O_S \mathit{QualityOfService} \otimes O_S \mathit{Replace3days}. \end{aligned}$$

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 contract cannot be taken in isolation in so far as they exist in a contract. Consequently the whole contract determines the meaning of each single clause 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 contract.

Another common case is exemplified by the rules:

$$r : \mathit{Invoice} \vdash O_P \mathit{PayWithin7Days} \otimes O_P \mathit{PayWithInterest}$$
$$r' : \mathit{Invoice}, \neg \mathit{PayWithin7Days} \vdash O_P \mathit{PayWithInterest}.$$

The first rule says that after the seller sends the invoice the buyer has one week to pay it, otherwise the buyer has to pay the principal plus the interest. Thus we have the primary obligation $O_P \mathit{PayWithin7Days}$, whose violation is repaired by the secondary obligation $O_P \mathit{PayWithInterest}$, while, according to the second rule, given the same set of circumstances *Invoice* and $\neg \mathit{PayWithin7Days}$ we have the primary obligation $O_P \mathit{PayWithInterest}$. 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 contract.

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

Definition 1 *Let* $r_1 : \Gamma \vdash A \otimes B \otimes C$ *and* $r_2 : \Delta \vdash D$ *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. $\Gamma \cup \{\neg A_1, \ldots, \neg A_m\} = \Delta$ *and* $D = B$; *or*
3. $\Gamma \cup \{\neg B_1, \ldots, \neg B_n\} = \Delta$ *and* $D = A \otimes \bigotimes_{i=0}^{k \leq p} C_i$.

The intuitions 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 arises in contracts. What we have to determine is whether we have genuine conflicts, i.e., the contracts is 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. FCL has facilities to detect conflicts and to resolve them. However, in this paper we are not interested in such features, and we will assume that a contract does not result in any conflict. For the details about how to detect conflicts see [13, 11]

5.1.3 Normalisation Process

We now describe how to use the machinery presented in Section 5.1.1 and Section 5.1.2 to obtain FCL normal forms. The FCL normal form of a contract provides a logical representation of a contract in format that can be used to monitor the execution of the contract. This consists of the following three steps:

1. Starting from a formal representation of the explicit clauses of a contract we generate all the implicit conditions that can be derived from the contract by applying the merging mechanism of FCL.
2. We can clean the resulting representation of the contract by throwing away 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 contract 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 [13], thus according to standard set theory results the fixed-point exists and it is unique. However, we have to be careful since merging first and doing subsumption after produces different results from the opposite order (i.e., subsumption first and merging after), or by interleaving the two operations.

5.2 Representing the Contract in FCL

Let us now see how to represent the contract of Section 3 in FCL. Usually a contract comprises two types of clauses: definitional clauses giving the meaning of the terms used in the contract and clauses specifying the normative behaviours (i.e., giving the obligations, permissions, prohibitions the signing parties of the contract are subject to). In this paper we will concentrate only on the normative specifications of a contract. Accordingly, we will ignore Sections 1, 2, and 3 of the contract, and similarly for clause 4.1 which states what the basic prices are and clause 6.2 that states what a payment is (see [9, 10], for a representation of these clauses in the spirit of an extension of FCL).

$r_{4.2}$: $\mathit{begin} \vdash O_P \mathit{FirstOrder7Days}$
$r_{5.1a}$: $\mathit{Service} \vdash O_S \mathit{QualityOfService}$
$r_{5.1b}$: $\neg \mathit{QualityOfService} \vdash O_S \mathit{Replace3days}$
$r_{5.1c}$: $\neg \mathit{Replace3ays} \vdash O_S \mathit{Refund\&Penalty}$
$r_{5.2}$: $\mathit{PurchaseOrder} \vdash O_S \mathit{Deliver1day}$
$r_{5.3a}$: $\mathit{Service}, \neg \mathit{QualityOfService}, \neg \mathit{Replace3days}, \neg \mathit{Refund\&Penalty} \vdash P_P \mathit{ChargeSupplier}$
$r_{5.3b}$: $\mathit{PurchaseOrder}, \neg \mathit{Deliver1day} \vdash P_P \mathit{ChargeSupplier}$
$r_{6.1}$: $\mathit{Invoice} \vdash O_P \mathit{PayWithin7days} \otimes O_P \mathit{PayWithInterest}$
$r_{7.1}$: $\mathit{3LatePayments} \vdash P_S \mathit{TerminateContract}$

The normalisation process give us the following rules

$r_{4.2}$: $\mathit{begin} \vdash O_P \mathit{FirstOrder7Days}$
$r_{5.1}$: $\mathit{Service} \vdash O_S \mathit{QualityOfService} \otimes O_S \mathit{Replace3days} \otimes O_S \mathit{Refund\&Penalty} \otimes P_P \mathit{ChargeSupplier}$
$r_{5.2}$: $\mathit{PurchaseOrder} \vdash O_S \mathit{Deliver1day} \otimes P_P \mathit{ChargeSupplier}$
$r_{6.1}$: $\mathit{Invoice} \vdash O_P \mathit{PayWithin7days} \otimes O_P \mathit{PayWithInterest}$
$r_{7.1}$: $\mathit{3LatePayments} \vdash P_S \mathit{TerminateContract}$

6 From FCL to business processes

In this section we describe how a business contract stated in the FCL can be translated into business processes that are aligned with contract conditions. We will attempt to refine contract description into increasing level of detail, starting from mere cross-organisational interactions and progressively refining it into internal processes. We will again use our contract example to illustrate the steps involved. Note that the example is not elaborated in full detail and many points were omitted for brevity.

We use BPMN notation [22] as our target process description language. We chose BPMN because this notation is suited to support business analysts and business process modellers and we believe that initial translation from business contracts into business processes needs to be undertaken by these specialists. Once this activity is performed and business process model is produced, it is up to the business process engine specialists to describe specific mapping from a BPMN model into an executable form of a process. Indeed, the BPMN specification comes with one such mapping, i.e. the mappings onto BPEL [5]. Many other mappings could be also possible.

In addition, BPMN provides an expressive, event-oriented approach to specifying contracts, which is a style of expression that is suitable for the description of business contracts and without imposing block-structure limitation adopted as part of BPEL specification.

Finally, BPMN provides quite 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.

In what follows, we will use all these process variants as facilities to help the progressive refinement from the contract onto processes.

6.1 From FCL to Abstract Processes

Consider the FCL representation of the contract given in the previous section. Our first step is to translate this contact representation into two abstract BPMN processes, corresponding to each of the parties and to identify the messages that can be consequences of the corresponding obligations. So, we first identify those obligations between the parties which explicitly state the subject and the target (or beneficiary) of the obligation. This is because many of such obligations will imply some form of message exchange between the parties. For example, the following statement

$$\neg \mathit{QualityOfService} \vdash O_S \mathit{Replace3Days}$$

implies that the Purchaser will need to send a message to the Supplier to inform them (about some problems). We can use BPMN text annotations as a way of stating deontic constraints on process diagrams. Note that we use the BPMN annotations to show triggering conditions for obligations, such as beginning of month trigger for sending monthly report. Thus, they are used to denote the conclusion and antecedents of the FCL logical expression respectively.

Note that the messages identified do not necessarily need to represent a unique or even best solution for the interactions between the parties. However, they can be a good starting point for subsequent redesign as needed.

6.2 From an abstract process to a private process

Once public processes are identified, it is possible to provide a more detailed structure of internal processes of the parties, including the way messages (identified in the abstract processes) are generated or consumed by the tasks within them.

The FCL expressions of the contract can again be used in this exercise. To this end, we have identified several translation rules, as follows.

First, it is useful to look for the antecedents of each of the deontic expressions that indicate actions of the parties involved in the contract. This is because antecedents represent certain occurrences, e.g. message arrivals, deadline expirations or state changes. In terms of BPMN constructs these would be start events or intermediate events, each of these of any of the BPMN available types. These triggers can then be used to start corresponding tasks of the internal processes. The statement $\mathit{Service} \vdash O_S\, \mathit{QualityOfService}$, for example, means that the if a service is requested the Supplier is obliged to provide it according to the published quality of service description.

Second, some of the tasks activated by the events, can in turn generate messages to be sent to the other party, for example $\mathit{begin} \vdash O_P \mathit{FirstOrder7Days}$ means that on the commencement of the contract the Purchaser has to send a message to the Supplier with the request for the first order.

Third, some obligations involve deadlines, and for this the triggering event should activate the tasks that need to be performed and also the deadline event, which if triggered before the completion of the activity, will signify violation of the obligation.

It is important to note that the rules identified above can be considered only as a guiding mechanism for identifying some patterns for structuring internal activities. There are of course many possible designs for internal business processes and this is where it may be useful to consider further heuristics. Some initial considerations for such heuristics are presented in [25].

6.3 From process description to process implementation

In the discussions so far we have been concerned with business level specification of processes. It is precisely for these reasons that we chose the BPMN notation - because it was developed for the high level descriptions of processes. However, in order to provide significant level of business process automation one relies on the availability of business process engines that provide an automation of business processes. In the case of BPMN the obvious engines of choice are those that implement BPEL semantics, because the BPMN specification provides mapping to the BPEL. However BPEL semantics provides a relatively restricted support for the execution of complex events along the lines of those that are for example described in [19]. It would be interesting to consider some other process engines that are more distributed in nature and that implement event-centric semantics such as those that follow the WS-CDL [26] approach or an engine proposed by Berry [3, 4].

6.4 A methodology for deriving compliant processes

Our approach for the progressive development of collaborative business processes from the legalese form of contract is part of a more general methodology which we have recently proposed [20]. The aim of the methodology is to serve as a general guiding tool for business process modellers in their activities of developing processes that are compliant with contracts. In doing so, they need to consider various business setups of organisations and various pre-existing procedures, policies and cultures. In some cases for example, when there are existing processes in place, there needs to be checked whether these processes are complaint with business contracts. In other cases, such as those addressed in this paper, one is concerned with how to design new processes based on the existing or new contract. In yet another cases one needs to determine whether there is sufficient trust between organisations to decide whether some third-party monitoring mechanisms need to be employed. All these issues need to be discussed and addressed when considering various deployment and operational issues associated with processes governed by business contracts.

7 Ideal Semantics

In a way, FCL constraint expressions for a contract define a behavioural and state space which can be used to analyse how well different behaviour execution paths (including state constraints) 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 contract. 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 FCL obligation rule related to our contract:

$$\mathit{WeekDay}, \mathit{FaultMessageEvent} \vdash O_S \mathit{Repair24hours}$$

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 *PremiumCustomer* (state variable) sending of two additional messages (an apology message, and a reparation confirmation message) which are not critical for the obligation.

Consider another example:

$$\mathit{WeekDay}, \mathit{PremiumCustomer}, \mathit{FaultMessageEvent} \vdash O_S \mathit{repair12hours}$$

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

$$\mathit{WeekDay}; \mathit{FaultMessageEvent}$$

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

7.1 FCL expressions and behavioural execution paths

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.

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 contract. A *sub-ideal* situation is situation where there are some violations, but these are repaired, in the CTD sense. Accordingly, processes resulting in sub-ideal situations are still compliant to a contract even if they provide non-optimal performances of the contract. A situation is *non-ideal* if it violates a contract (and the violations are not repaired). In this case a process resulting in a non-ideal situation does not comply with the contract. There are two possible reasons for a process not to comply with a contract: 1) the process executes some tasks which are prohibited by the contract (or equivalently, it executes the opposite of obligatory tasks); 2) the process fails to execute some tasks required by the contract. Finally a situation is irrelevant for a contract if no rule is applicable in the situation. Irrelevant situations correspond to states of affairs where a contract is silent about them.

In the rest of this section we provide a formal definition for these concepts.

As discussed in Section 5.1, for every FCL representation of a contract its normal form contains all conditions that can be derived from the contract and redundant clauses are removed. Thus normal forms are the most appropriate means to determine whether a process conforms with a contract. Accordingly, we have to use the normal form of a contract and not the contract itself to determine whether a business process complies with the contract. We now define conditions under which we are able to determine whether a situation complies with a contract or if it represents a violation of some clauses. To this end, we shall define a *situation* to be a pair (L, S) where L is a set of literals representing states and S is a pattern (sequence) of events.

In what follows we will consider the rules in the normal form for a contract. In addition every FCL rule

$$B_1, \ldots, B_m \vdash A_1 \otimes \ldots \otimes A_n$$

will be represented as

$$\Gamma, E \vdash A_1 \otimes \ldots \otimes A_n$$

where Γ is the set of state literals in $\{B_1, \ldots, B_m\}$ and E is the conjunction of the event literals in $\{B_1, \ldots, B_m\}$, and $1 \leq n$. For example, given the rule

$$\mathit{WeekDay}, \mathit{PremiumCustomer}, \mathit{FaultMessageEvent}, \mathit{RequestOnSite} \vdash O_S \mathit{SendTechnician}$$

we have that

$$\Gamma = \{\mathit{WeekDay}, \mathit{PremiumCustomer}\}$$
$$E = \mathit{FaultMessageEvent} \wedge \mathit{RequestOnSite}$$

Definition 2 *Given two sequences of events S and S', we say that S' is a subsequence of S, if every element of S' is an element of S, and the elements of S' occur in the same order as they occur in S.*

For example, given the sequence of events $S = \alpha; \beta; \gamma; \delta; \epsilon$ the sequence $\beta; \epsilon$ is a subsequence of S but $\gamma; \beta$ is not since γ occurs before β in $\gamma; \beta$ while it occurs after β in S.

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

Definition 3

- *A situation $s = (L, S)$ is* ideal *with respect to a rule $\Gamma, E \vdash A_1 \otimes \cdots \otimes A_n$ iff if $\Gamma \subseteq L$ and E is a subsequence of S, then $E; A_1$ is a subsequence of S.*
- *A situation $s = (L, S)$ is* sub-ideal *with respect to a rule $\Gamma, E \vdash A_1 \otimes \cdots \otimes A_n$ iff if $\Gamma \subseteq L$, E is a subsequence of S and $\exists A_i$, $1 < i \leq n$ such that $\forall A_j$, $j < i$ $E; \neg A_1^+; \ldots; \neg A_j^+; A_i$ is a subsequence of S.*[2]
- *A situation $s = (L, S)$ is* non-ideal *with respect to a rule $\Gamma, E \vdash A_1 \otimes \cdots \otimes A_n$ iff $\Gamma \subseteq L$ and E is a subsequence of S and s is neither ideal nor sub-ideal.*
- *A situation $s = (L, S)$ is* irrelevant *with respect to a rule $\Gamma, E \vdash A_1 \otimes \cdots \otimes A_n$ iff it is neither ideal nor sub-ideal nor non-ideal.*

Returning to our first example, $\Gamma = \{\mathit{WeekDay}\}$ and the sequence of events $E = \mathit{FaultMessageEvent}$ L is $\{\mathit{WeekDay}, \mathit{PremiumCustomer}\}$ and the sequence of events S is

$$S = \mathit{FaultMessageEvent}; \mathit{SendApologyMessage};$$
$$\mathit{Repair24hours}; \mathit{SendReparationConfirmationMessage}$$

So, it is true that $\Gamma \subseteq L$, and E and $E; \mathit{Repair24hours}$ are subsequences of S.

According to Definition 3, 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 3 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 contract containing all rules inherent to the contract.

Definition 4

[2]With $\neg A_k^+$ we denote 0 or 1 occurrence of $\neg A_k$ in a sequence of events.

- *A situation s is* ideal *with respect to a contract normal form iff there is no rule in the normal form for which s is either sub-ideal or non-ideal or irrelevant.*
- *A situation s is* sub-ideal *with respect to a contract normal form iff there is a rule for which s is not irrelevant and it is sub-ideal, and there is no norm in the normal form for which s is non-ideal.*
- *A situation s is* non-ideal *with respect to a contract normal form iff there is no rule in the normal form for which s is not irrelevant and is non-ideal.*
- *A situation s is* irrelevant *with respect to a contract normal form iff for all rules in the normal form s is irrelevant.*

Definition 4 follows immediately from the intuitive interpretation of ideality and the related notions we have provided in Definition 3. On the other hand, the relation between a normal form and the contract from which it is obtained seems to be a more delicate matter. A careful analysis of the conditions for constructing a contract normal form allows us to state the following general criterion:

Definition 5 *A situation s is ideal (sub-ideal, non-ideal, irrelevant) with respect to a contract FCL if s is ideal (sub-ideal, non-ideal, irrelevant) with respect to the contract normal form from FCL.*

It is worth noting that Definition 5 shows the relevance of the distinction between a contract and its normal form. This holds in particular for the case of sub-ideal situations. Suppose you have an FCL contract consisting of the rules

$$\vdash O_s A \qquad \neg A \vdash O_s B$$

The corresponding contract normal form is

$$\vdash O_s A \otimes O_s B$$

While the situation with $\neg A; B$ is sub-ideal with respect to the latter, it would be non-ideal for the former. In the first case, even if $\neg A \vdash B$ expresses in fact an implicit reparational obligation of $\vdash A$, this is not made explicit. Key point here is that there was no link between the primary and reparation obligations in the contract, 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 $\vdash A$ and $\neg A \vdash B$ must lead to a unique regulation. For this reason, we can evaluate a situation as sub-ideal with respect to an FCL contract only if it is sub-ideal with respect to its normal form.

7.2 Processes as Behaviour Execution Paths

In this section we treat business process fragments in terms of behaviour execution paths. This is a generic mechanism we can use to check compliance between business processes and business contracts.

As we have argued before there are two ways in which a process does not comply with a contract (or a contract rule).

1. It explicitly violates an obligation;
2. If fails to perform a required task.

For example consider the rule

$$\alpha \vdash O_s \beta \otimes O_s \gamma$$

which means that, if event α occurred then this must be followed by β, or in alternative, in case β does no occur, it must be followed by γ.

According to the intuitive reading and Definition 3, a situation is

- ideal if it has $\alpha; \beta$ as its subsequence;
- sub-ideal if it has either of the following as subsequences $\alpha; \neg\beta; \gamma$, $\alpha; \gamma$.
- not-ideal if it has α as a subsequence, but there is non subsequence extending α that has β or γ as its members.

Let us now consider the process

$$\pi = \delta; \alpha; \epsilon.$$

This process results in a non-ideal situation: it has α as one of its subsequences, but it does not contain β or γ. So it not compliant because it fails to fulfil the obligation $O_s\beta$.

The process[3]

$$\pi' = \delta; \alpha; \epsilon; \neg\beta$$

is also not compliant because in this case it presents an explicit violation of the obligation $O_s\beta$. The main difference between π and π' is that π can be made (fully) compliant by extending it with β, while π' either is revised by first removing $\neg\beta$ and then inserting a sub-process corresponding to β, or resulting in a sub-ideal situation by extending it by γ.

7.3 Ideal Semantics for the Contract

In Section 5.2 we have shown the FCL representation of the contract an the resulting normal form. Here we describe the minimal behaviours corresponding to ideal, sub-ideal and non-ideal situation for the normal form of the contract.

The minimal non-trivial ideal situation for rule $r_{4.2}$ is when $L = \emptyset$ and

$$S_1 = \mathit{begin}; \mathit{FirstOrder7Days}$$

and non-ideal if

$$S_2 = \mathit{begin}; \neg\mathit{FirstOrder7Days}$$

For rule $r_{5.1}$ the minimal non-trivial ideal situation is when $L = \emptyset$ and

$$S_3 = \mathit{Service}; \mathit{QualityOfService}$$

while the situations where S is either of the following

$$S_4 = \mathit{Service}; \neg\mathit{QualityOfService}; \mathit{Replace3days}$$
$$S_5 = \mathit{Service}; \neg\mathit{QualityOfService}; \neg\mathit{Replace3days}; \mathit{Refund\&Penalty}$$

are sub-ideal situations because there are events that must be executed in case of violations of prior obligations; while the following

$$S_6 = \mathit{Service}; \neg\mathit{QualityOfService}; \neg\mathit{Replace3days}; \mathit{Refund}$$
$$S_7 = \mathit{Service}; \neg\mathit{QualityOfService}; \neg\mathit{Replace3days}; \mathit{Penalty}$$

are non-ideal situations, since the last possible reparation (*Refund&Penalty*) is not completely fulfilled.

For $r_{5.2}$ we have

$$S_8 = \mathit{PurchaseOrder}; \mathit{Deliver1day}$$

is the minimal non-trivial ideal situation, while

$$S_9 = \mathit{PurchaseOrder}; \neg\mathit{Deliver1day}$$

is non-ideal.

Finally for $r_{6.1}$, the minimal non trivial situation is when

$$S_{10} = \mathit{Invoice}; \mathit{PayWithin7days}$$

while

$$S_{11} = \mathit{Invoice}; \neg\mathit{PayWithin7days}; \mathit{PayWithInterest}$$

is sub-ideal, and

$$S_{12} = \mathit{Invoice}; \neg\mathit{PayWithin7days}; \neg\mathit{PayWithInterest}$$

is not ideal, and then it does not comply with the contract.

[3]Notice that here we ignore the distinction whether β must immediately follow α, or just follows it. This this distinction can be made more precise with the introduction of temporal notions either as timestamps or based on intervals, for example using Allen's interval algebra [1, 2] as in [18]. However, this distinction, while important for properly representing business process, is not essential to the discussion of the present paper, since the argument will carry over unchanged to those more complex and powerful formalisms.

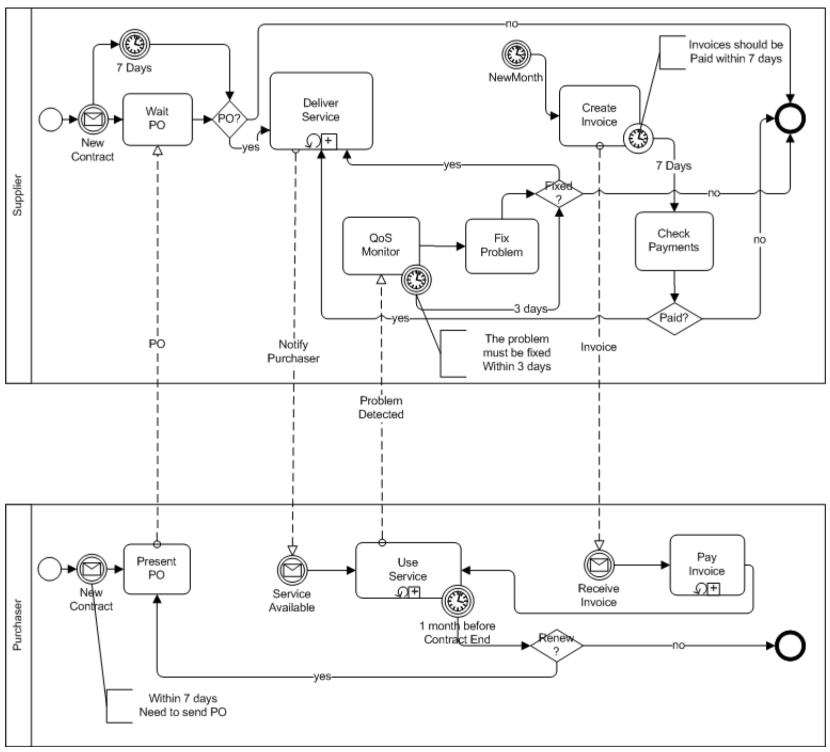

Figure 2: BPMN Diagram for Processes in Service Contract

8 Compliance of Processes with Contracts

In the previous discussion we have outlined some methodology and heuristic to derive business processes compliant with a business contract starting from the formal representation of the business contract. In this section the focus in instead in checking whether an existing business process is compliant with the relevant regulatory documents (contracts). As in the previous section, we use BPMN for the representation of business processes.

8.1 BPMN Formulation: The Service Contract

Let us assume that a business analyst has come up with a process representation as shown in Figure 2, independently of the details of the contract conditions. In other words there was no prior attempt to derive contract compliant processes as proposed in Section 6. In the figure key obligations are stated as BPMN annotations next to the actions that are expected to occur. We also use several timers that reflect temporal constraints on when the obligations should be discharged. The mapping from this process into a process event description follows the rules below:

- A BPMN intermediate event representing message arrival is a process event, e.g., a message signifying that new contract was signed, $e_{NewContract}$;
- A BPMN intermediate event representing timer is a process event;
- A BPMN message flow's origin is a process event; note that in BPMN both sending and receiving of a message is an event, but to simplify presentation we only regarded the sending of a message as a process event;
- Start of a BPMN task or process is a process event;
- The completion of a BPMN task or process is a process event;
- BPMN does not specify when a message flow is to be triggered from a task, so we will assume that it is triggered immediately following the start of this task (e.g., as in *DeliverService* and *CreateInvoice* tasks), unless the message is created when other change state occurs within this task, as in *ProblemDetected* message sent out of the *UseService* sub-process; this event will be sent when its corresponding guard in an process event description will be set to true.

These events can be combined into more complex structures. We use three simple operators to produce a simple event language used only for the purpose of illustrating the mapping from contracts to processes. We note that a more detailed mapping from BPMN into a more sophisticated event language is subject of our other research topic. Three simple event operators are used to describe their temporal causality (i.e., sequence), depicted with the ; symbol, their alternative paths (OR) or they simultaneous triggering (AND).

We assumed that a business modeller using BPMN needed a way to describe a common requirement that some task must be completed within a particular period of time, and if this is not the case then an alternative execution path need to be taken. There is currently no notation for this scenario. In fact, this can be regarded as a common pattern that can be modelled by using the BPMN intermediate timer event and a decision node together with the required task. So, it is upon the activation of the task in question that the timer is also activated, which will generate another event (deadline expiry). If that event occurs before the completion of the task, then the obligation in question is violated and this can be a sub-ideal or non-ideal situation. There are several usages of this pattern in Figure 2. For example *FixProblem* task is activated at the same time as the corresponding timer (3Days), and the flows from that timer and from the task are fed into an OR merging node, so that whichever event occurs first it would continue onto the subsequent flows via the merger node. We will use the name of the event and the name of the timer to denote the resulting complex event, in this case the event is *FixProblem3Days*.

8.2 Event-centric transformation

The example of Figure 2 is shown in terms of event patterns as follows.

Supplier Side: There are three event sub-patterns, which describe independently triggered execution paths associated with:

- the activity of waiting for Purchase Orders (and if received, starting delivering service)
- Invoicing activities
- Reaction to the problems

These patterns are listed next.

$\pi_1 = e_{NewContract}; WaitPO7days; ((POrecieved; DeliverService; m_{NotifyPurchaser}) \lor End)$
$\pi_2 = NewMonth; CreateInvoice; (m_{Invoice} \land (CheckPayment7days;$
$([Paid]DeliverService \lor [NotPaid]; End)))$
$\pi_3 = m_{ProblemDetected}; QoSMonitor; FixProblem3days; ([fixed]DeliverService \lor [notFixed]End)$

Notice that these patterns will need to be composed into one high-level pattern that will represent execution path options for the overall process. So, these three sub-patterns are then AND-ed and *NewMonth* and $m_{ProblemDetected}$ can then become part of guards for the last two patterns.

Similar reasoning applies to the Purchaser Side, and the three patterns there are:

$\pi_4 = e_{NewContract}; PresentPO; m_{PO}$
$\pi_5 = m_{NotifyPurchaser}; UseService; ([problem]m_{ProblemDetected} \land D_{1monthBeforeContractEnd});$
$([Renew]PresentPO \lor [NoRenew]End)$
$\pi_6 = m_{RecieveInvoice}; PayInvoice; UseService$

8.3 ANDs and ORs

The event relationships from the BPMN example above include several operators that were not included in the execution path expressions derived from the normal form of the contract expressed in FCL. These are the AND and OR operators that can describe two separate branches in a process, and Guards which can be likened to the states in the FCL antecedents. The lack of such operators within the expressions of behaviour execution paths results from the limitations of current behaviour execution paths formalism which is primarily influenced by the sequence relationship between antecedent and conclusion in the FCL statements.

Although we do plan to extend this execution path formalism (for example to express Obligation on two events in AND relationship) this limitation is not the problem for this example because the Ideal semantics can incorporate compliance checking for the AND and OR branches. It does so by providing a union of execution paths from AND or OR branches.

In case of OR branches an ideal situation for the whole process is satisfied if any of the branches are ideal.

In case of AND branches, if all AND branches are ideal, then the composite process is ideal. However, if any of AND branches is non-ideal or sub-ideal so is the overall process. This area needs further investigation.

8.4 BPMN Compliance with FCL

We are now ready to test whether the specifications of the path obtained from the BPMN diagram comply with the FCL representation of the contract according to the ideal semantics presented in Section 7.3.

Process π_1 contains an OR-branch, thus it generates two sequences

$$\begin{aligned} P_1 = {} & e_{NewContract}; \mathit{WaitPO7days}; \mathit{POrecieved}; \mathit{DeliverService}; m_{NotifyPurchaser} \\ P_2 = {} & e_{NewContract}; \mathit{WaitPO7days}; \mathit{End} \end{aligned}$$

Assuming that the event *POrecieved* corresponds to the event *PurchaseOrder*, and event *DeliverService* to *Deliver*, then the path P_1 does not comply with rule $r_{5.2}$ since it lacks the timer $1day$. Path P_2 does not contain any subsequence of the sequences given in Section 7.3, and thus it is deemed as irrelevant. We can argument in the same way (lack of timer) to determine the non compliance of processes π_4 and π_5

Similarly to the previous case process π_3 generates two paths:

$$\begin{aligned} P_3 &= m_{ProblemDetected}; \mathit{QoSMonitor}; \mathit{FixProblem3days}; [\mathit{fixed}]\mathit{DeliverService} \\ P_4 &= m_{ProblemDetected}; \mathit{QoSMonitor}; \mathit{FixProblem3days}; [\mathit{notFixed}]\mathit{DeliverService} \end{aligned}$$

Again assuming that the same ontology is used for event literals in FCL and event patterns in BPMN, $\neg$*QualityOfService* maps to $m_{ProblemDetected}$ and *Replace* to *FixProblem*, then P_3 corresponds to a sub-ideal situation (see sequence S_4 in Section 7.3) for rule $r_{5.1}$, In this case *Replace* and *FixProblem* have the same timer. On the other hand, P_4 is not-ideal since the process repairing the violation of the then required event *FixProblem3days* is that the supplier has to refund the customer and to pay her a penalty, and not the termination of the contract.

Finally π_2 and π_5 are not relevant for the contract (i.e., they do not affect whether the whole processes complies or not with the contract).

9 Conclusions and Future work

In this paper we have embarked on a relatively unexplored research theme related to compatibility checking of business processes against business contracts. The value of this research lies in the need to address a number of incompatibility problems in the business world, resulting from varying business environments, characterised by different business setups, external constraints as well as future organisations trajectories.

One specific compatibility area that is addressed in this paper is concerned with ensuring compliance between business processes and business contracts. This is of relevance for many organisational environments of today in which business contracts and business processes are designed and managed through separate activities and by separate specialists. This can lead to compliance problems that in turn may lead to the violation of contract conditions with possibly costly consequences both in financial and reputation terms.

Our approach to compliance checking involves two solution approaches. The first solution provides a set of guidelines for progressively transforming contract conditions into business processes that are consistent with contract conditions thus avoiding violation of the rules in contract. Another solution compares rules in business contracts and rules in business processes to check for possible inconsistencies. Both solutions use a logic-based formalism for the expression of contracts and their violations, coupled with a new semantics that we have developed specifically for the purpose of compliance checking. The formalisation employs an event-based way of expressing behaviour associated with contracts and processes. The semantics consists of determining what are the ideal, sub-ideal and non-ideal situations (or state of affairs) when comparing business process execution paths and contract conditions. We have tested our approach by assuming that processes and contracts are developed by different experts and we have found several compliance problems in the example used, as reported.

This exercise has also identified several research issues that we intend to address in future regarding compliance checking.

For the derivation of compliant processes from normative specifications we have found that the representation of contract in a structured form such as the one based on a logic formalism, provides a more direct way in the identification of messages to be exchanged between parties as driven by their mutual obligations - as opposed by using straight legalese version. We have also found that FCL form can help in identifying some fragments of internal processes, typically the tasks that are activated by some triggering events and tasks which need to be involved in exchange of messages needed as part of fulfilment of mutual obligations. However, it is difficult to arrive at a detailed set of rules for the construction of internal processes - which in a way is to be expected as there may be many ways how processes can be realised to implement contract conditions. Some heuristics can be applied in a similar way as was proposed in [20] and we expect that more of such heuristics will be developed over time as the patterns of use are identified and recorded. In fact, one of our future research directions to study various such cases and to provide richer set of heuristics, which are likely to be industry domain dependent.

For the other solution approach, our current semantics supports relatively simple normative expressions in which deontic constraints are expressed in terms of single events. We plan to extend it by considering complex event relationships instead, e.g., obligations involving two of more events related according to various compositional operators. In addition, we plan to investigate broader context for the Ideal semantics and cater for a more complex compositional operators (than sequence only) between events in behaviour execution paths, for example as in [27]. Further, our example has pointed to a need to provide a better handling of deadlines in FCL Obligation modalities, because they are frequently used in the specification of obligations. In particular we will try to incorporate in FCL the approaches of [14] to represent temporal notions and temporalised normative positions and of [6] to incorporate obligation deadlines.

The example has also identified a need to establish a common ontology for the events that are used in contract and process specifications and we intend to investigate possible solution approaches for this area.

Another topic is related to checking of a different form of compatibility, namely the one which involves an additional checking of possible future resource conflicts. For example, are partners's future commitments such that they may lead to the violations of the contract in question.

Acknowledgements

This work is partially supported by the Australia Research Council under Discovery Project No. DP0558854 and Linkage Project No. LP0348532.

References

[1] James F. Allen. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154, 1984.

[2] James F. Allen and George Ferguson. Actions and events in interval temporal logic. *J. Log. Comput.*, 4(5):531–579, 1994.

[3] Andrew Berry. *Describing ans Supporting Complex Interactions in Distibuted Systems.* Phd thesis, University of Queensland, 2002.

[4] Andrew Berry and Zoran Milosevic. Extending choreography with contract constraints. *International Journal of Cooperative Information Systems*, 14(2–3):131–179, 2005.

[5] Business Process Execution Language for Web Services. Version 1.1. `http://www-128.ibm.com/developerworks/library/specification/ws-bpel/`, 5 May 2003.

[6] Jan Broersen, Frank Dignum, Virginia Dignum, and John-Jules Ch. Meyer. Designing a deontic logic of deadlines. In Alessio Lomuscio and Donald Nute, editors, *Deontic Logic in Computer Science*, volume 3065 of *Lecture Notes in Computer Science*, pages 43–56. Springer, 2004.

[7] José Carmo and Andrew J.I. Jones. Deontic logic and contrary to duties. In D.M. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 8, pages 265–343. Kluwer, Dordrecht, 2002.

[8] Elkera XML-2-go suite. http://www.elkera.com.

[9] Guido Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2–3):181–216, 2005.

[10] Guido Governatori and Duy Hoang Pham. A semantic web based architecture for e-contracts in defeasible logic. In Asaf Adi, Suzette Stoutenburg, and Said Tabet, editors, *RuleML*, volume 3791 of *Lecture Notes in Computer Science*, pages 145–159. Springer, 2005.

[11] Guido Governatori and Zoran Milosevic. Dealing with contract violations: formalism and domain specific language. In *9th International Enterprise Distributed Object Computing Conference (EDOC 2005)*, pages 46–57. IEEE Computer Society, 2005.

[12] Guido Governatori and Antonino Rotolo. A Gentzen system for reasoning with contrary-to-duty obligations. A preliminary study. In Andrew J.I. Jones and John Horty, editors, *Δeon'02*, pages 97–116, London, May 2002. Imperial College.

[13] Guido Governatori and Antonino Rotolo. Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic*, 4:193–215, 2006.

[14] Guido Governatori, Antonino Rotolo, and Giovanni Sartor. Temporalised normative positions in defeasible logic. In Anne Gardner, editor, *10th International Conference on Artificial Intelligence and Law (ICAIL05)*, pages 25–34. ACM Press, 2005.

[15] Gartner Group. *Delivering ITs Contribution: The 2005 CIO Agenda.* Gartner, Inc, Stamford, Connecticut, 2005.

[16] Andrew .J.I. Jones and Marek Sergot. A formal characterisation of institutionalised power. *Journal of IGPL*, 3:427–443, 1996.

[17] Stig Kanger. Law and logic. *Theoria*, 38:105–32, 1972.

[18] Ruopeng Lu, Shazia Sadiq, Vineet Padmanabhan, and Guido Governatori. Using a temporal constraint network for business process execution. In Gillian Dobbie and James Bailey, editors, *Seventeenth Australasian Database Conference (ADC2006)*, volume 49 of *CRPIT*, pages 157–166, ACS, 2006.

[19] David Luckham. *The Power of Events.* Addison-Wesley, 2002.

[20] Zoran Milosevic, Shazia Sadiq, and Maria Orlowska. Towards a methodology for deriving contract-compliant business processes. In *Proc. 4th Int. Conference on Business Process Management*, pages 395–400. Springer, 2006.

[21] OASIS. LegalXML eContracts. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=legalxml-econtracts.

[22] Object Management Group. Business Process Modeling Notation Specification. http://www.bpmn.org/, 3 February 2006.

[23] Sarbanes-Oxley Act of 2002. US Public Law 107-204. http://www.sec.gov/about/laws/soa2002.pdf.

[24] Health Insurance Portability and Accountability Act of 1996. US Public Law 104-191.

[25] Roger Tagg, Zoran Milosevic, Sachin Kulkarni, and Simon Gibson. Supporting contract execution through recommended workflows. In Fernando Galindo, Makoto Takizawa, and Roland Traunmüller, editors, *Database and Expert Systems Applications, 15th International Conference, DEXA 2004*, volume 3180 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2004.

[26] Web Service Choreography Decription Language, 1.0. www.elkera.com, 30 November 2006.

[27] Adam Zachary Wyner. A functional program for agents, actions, and deontic specifications. In Matteo Baldoni and Ulle Endriss, editors, *Proceedings of DALT 2006*, 2006.

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