Compliance checking between business processes and business contracts

Guido Governatori*, Zoran Milosevic*† and Shazia Sadiq*
*School of ITEE, The University of Queensland, Brisbane, QLD 4072, Queensland, Australia
Email: {guido,zoran,shazia}@itee.uq.edu.au
†Deontik, Australia, Email: zoran@deontik.com

Abstract— **It is a typical scenario that many organisations have their business processes specified independently of their business contracts. 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 paper provides 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 central part of the paper are logic based formalism for describing both the semantics of contract and the semantics of compliance checking procedures.**

I. Introduction

Many organisations are striving to automate their business processes to improve their internal efficiency but they have different levels of awareness about the impact of external constraints on their processes. The constraints can arise from various regulatory and legislative requirements, including international trade agreements, but also from business contracts that regulate cross-organisational interactions. This paper considers constraints arising from business contracts and the implications of such limited awareness as a factor that has led to different levels of automation of business contract and business processes and often their stark separation.

It is this state of affairs, but also different business life cycle of business contracts and business processes, as well as different domain knowledge of experts constructing them, that requires a generic approach to checking compatibility of business contracts and business processes. This compatibility will be of even higher importance in future in the likely case of organisations wishing to rely more on enterprise contract management systems as a way of more direct integration of contracts and business processes and flexibility in changes that can propagate both directions.

In order to motivate the need for such compatibility checking, we consider two distinct scenarios that correspond to two ends of business contract automation spectrum. We describe some implications of these on incompatibility between the two sub-systems.

The most typical scenario today is one in which organisations have substantial separation of contracts from business processes because they still treat contracts as legal documents detached from their governance role for cross-organisational processes. The contract management activities are mostly manual and the only connection can be certain specialised contract management roles that look after contracts and occasionally have insight into the details of business processes. In this case there is no incompatibility between automation aspects (because of the mere lack of electronic support for contracts) but there is incompatibility in terms of efficiency and effectiveness between these two different sub-systems. If such organisations consider automation of their contracts and aim at ensuring compliance between contract conditions and processes structure, they need to have a mechanism for understanding impact of the existing business processes on new contracts. The impact can be in terms of rules defining best practices currently implemented or in terms of their future resource commitments, maybe as part of some other contracts. So, before an organisation agrees on new contracts, it would benefit from checking whether the desired contract conditions are compatible with existing processes, to avoid undesired consequences of inability to fulfil its obligations.

On the other end of the spectrum is scenario covering the most sophisticated support for business contracts, provided through a state-of-art enterprise contract management system, or as part of organisations ERP systems. These systems support contract authoring and various kind of contract monitoring, but they rarely provide capability to derive business processes in a way that is compliant with business contract. This is perhaps due to inherent difficulty of such derivation arising from different style of behavioural descriptions associated with contracts and processes, and their different purpose, as discussed in [14]. It is evident that this scenario also requires a mechanism to facilitate compatibility checking between processes and contracts. In fact, this mechanism can be regarded as part of a general methodology for deriving processes from contracts.

In summary, each of these scenarios requires a mechanism to facilitate better analysis of mutual dependencies between constraints stated in contracts and business process representation. This paper presents an approach for checking such dependences and ensuring compatibility between contracts and processes. This approach exploits our previous results related to the formal expressions of business contracts and our previous involvement in business process research and standardisation. We use our earlier example of business contracts presented in [7], to facilitate presentation and continuity of idea development.

In the next section we introduce the contract used a refer-

Proceedings of The 10th International Enterprise Distributed Object Computing Conference (EDOC 2006), pp. 221–232.

The original publication is available at doi: 10.1109/EDOC.2006.22

ence to illustrate the notions we are going to discuss in the paper. In Section III we summarise key concepts and rules of a Formal Contract Language (FCL) and show how FCL expressions can reduce to normal forms (Section IV) and we provide a semantics for it. We then introduce an example of a business process, intended to support the reference contract. We use standard BPMN notation to represent business process and then derive and an event-based specification for this process (Section VI). This intermediate representation was needed to facilitate mapping of contract conditions on behaviour patterns in the business process. The semantics of this mapping is then shown.

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.

II. A Sample Contract

This section introduces an example of a business contract, initially introduced in [7]. 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.

III. FCL

We now introduce the logic (FCL) we will use to reason about contracts. FCL was introduced in [7] for the formal analysis of business contracts and it is based on previous work on formal representation of contracts [5], logic of violations [8], and normative positions based on Deontic Logic with Directed Obligations [11], [10]. 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 $Ol_1, \dots, Ol_n$ are modal literals and l_{n+1} is a literal, then $Ol_1 \otimes \dots \otimes Ol_n$ and $Ol_1 \otimes \dots \otimes Ol_n \otimes Pl_{n+1}$ 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.

IV. 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 IV-A) 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 IV-B, we examine the problem of redundancies, and we give a condition to identify and remove redundancies from the formal specification of a contract.

A. 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:

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

This reads as follows: given two policies such that one is a conditional obligation ($\Gamma \vdash O_sA$) 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., $\Gamma, \Delta \vdash O_sA \otimes O_{s'}C$) 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

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

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.

$$\frac{\Gamma \vdash O_sA \otimes O_sB \qquad \neg A, \neg B \vdash O_sC}{\Gamma \vdash O_sA \otimes O_sB \otimes O_sC}$$

[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.

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

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

from which we derive the new rule

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

The above patterns are just special instances of the general mechanism described in details in [8], [5].

B. 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

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

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 $\mathit{QualityOfService}$ 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:

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

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

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 [8], [7]

C. FCL Normal Forms

We now apply the logical machinery presented to validate and transform business contracts into the logical representation in a language apt to monitor the execution of a 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 [8], 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.

D. Representing the Contract in FCL

Let us now see how to represent the contract of Section II 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 [5], [6], 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

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

V. 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.

A. 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 IV, 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

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

we have that

$$\begin{aligned} \Gamma &= \{\mathit{WeekDay}, \mathit{PremiumCustomer}\} \\ E &= \mathit{FaultMessageEvent} \wedge \mathit{RequestOnSite} \end{aligned}$$

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]

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

- *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 $E = \mathit{FaultMessageEvent}; \mathit{Repair24hours}$ L is $\{\mathit{WeekDay}, \mathit{PremiumCustomer}\}$ and the sequence of events S is

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

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

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

- *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 it 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.

B. 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 has a subsequence, but no subsequence extending α 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]

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

[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 [12]. 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.

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.

C. Ideal Semantics for the Contract

In Section IV-D 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

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

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

$$\begin{aligned} S_6 &= \mathit{Service}; \neg\mathit{QualityOfService}; \neg\mathit{Replace3days}; \mathit{Refund} \\ S_7 &= \mathit{Service}; \neg\mathit{QualityOfService}; \neg\mathit{Replace3days}; \mathit{Penalty} \end{aligned}$$

are non-ideal situations, since the last possible reparation ($\mathit{Refund\&Penalty}$) 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.

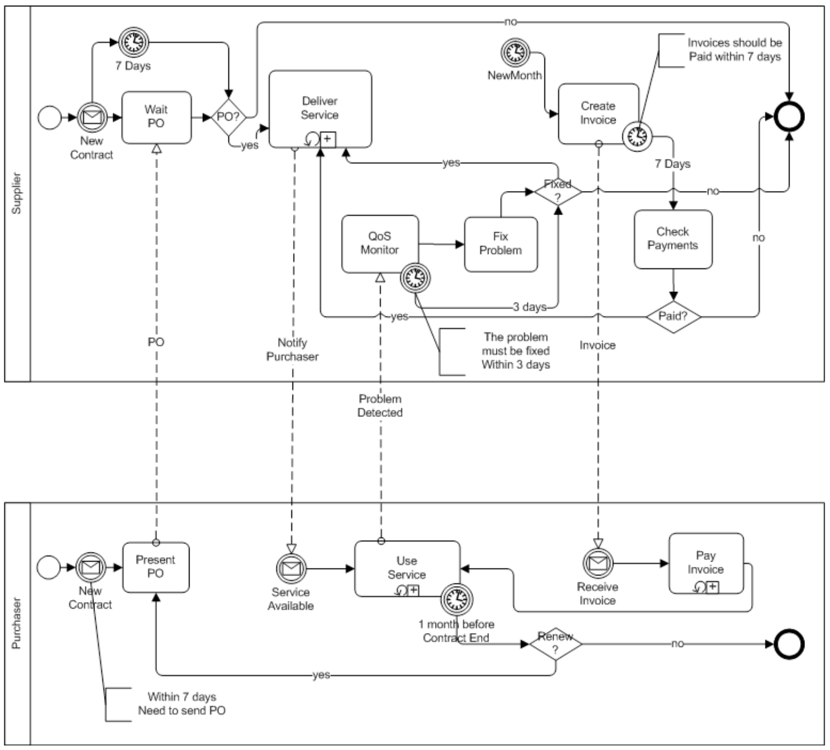

Fig. 1. BPMN Diagram for Processes in Service Contract

VI. Processes and Contracts

This section presents one specific implementation of a business process that implements key behaviour fragments of the service contract. We will assume that this model captures the key interactions between Supplier and Purchaser and some internal interactions within each of the parties as they are currently implemented. Our aim is to determine compliance of a business process with a contract.

In order to facilitate compliance checking we will aim to express business process in a form that is similar in style to the formalism of business contract, i.e. in terms of event-oriented behaviour. We begin first by expressing this business process using BPMN notation. We chose BPMN [13] for two reasons. Firstly, it comes with a number of concepts that are suitable for the expression of both cross-organisational and internal business processes, both of which are needed when refining contract behaviours. Second, BPMN is well suited for the use of business analysts and other business people involved in managing and monitoring processes. It is our belief that these kind of experts will be producing business process representation for the existing processes. Besides, this is a similar level of abstraction to that used in specifying business contracts.

It is important to state however, that BPMN notation requires additional specification details to be interpreted by underlying process engines. While the current version of BPMN specification provides such details through the inclusion of mapping rules between BPMN and BPEL [3] concepts, in this paper we provide added details by mapping the BPMN specification onto an event pattern language. Recall that this target was selected because the specification of contract condition is done in terms of event conditions and since both the source and target languages are event oriented, the mappings between these are more straightforward.

A. BPNM Formulation: The Service Contract

Let us assume that a business analyst has come up with a process representation as shown in Figure 1. In the figure key obligations are stated as BPMN annotations next to the actions that are expected to occur. We also use a number of 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 flows 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 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 $\mathit{DeliverService}$ and $\mathit{CreateInvoice}$ tasks), unless the message is created when other change state occurs within this task, as in $\mathit{ProblemDetected}$ message sent out of the $\mathit{UseService}$ 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 1. For example $\mathit{FixProblem}$ 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 $\mathit{FixProblem3Days}$.

B. Event-centric transformation

The example of Figure 1 is shown in terms of event patterns as follows.

a) 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.

$$\begin{aligned}
\pi_1 = {} & e_{NewContract}; \mathit{WaitPO7days}; \\
& ((\mathit{POrecieved}; \mathit{DeliverService}; m_{NotifyPurchaser}) \\
& \vee \mathit{End}) \\
\pi_2 = {} & \mathit{NewMonth}; \mathit{CreateInvoice}; \\
& (m_{Invoice} \wedge (\mathit{CheckPayment7days}; \\
& \qquad ([\mathit{Paid}]\mathit{DeliverService} \vee [\mathit{NotPaid}]; \mathit{End}))) \\
\pi_3 = {} & m_{ProblemDetected}; \mathit{QoSMonitor}; \mathit{FixProblem3days}; \\
& ([\mathit{fixed}]\mathit{DeliverService} \vee [\mathit{notFixed}]\mathit{End})
\end{aligned}$$

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 $\mathit{NewMonth}$ 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:

$$\begin{aligned}
\pi_4 = {} & e_{NewContract}; \mathit{PresentPO}; m_{PO} \\
\pi_5 = {} & m_{NotifyPurchaser}; \mathit{UseService}; \\
& ([\mathit{problem}]m_{ProblemDetected} \wedge D_{1monthBeforeContractEnd}); \\
& ([\mathit{Renew}]\mathit{PresentPO} \vee [\mathit{NoRenew}]\mathit{End}) \\
\pi_6 = {} & m_{RecieveInvoice}; \mathit{PayInvoice}; \mathit{UseService}
\end{aligned}$$

C. 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.

D. 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 V-C.

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 $\mathit{POrecieved}$ corresponds to the event $\mathit{PurchaseOrder}$, and event $\mathit{DeliverService}$ to $\mathit{Deliver}$, 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 V-C, 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:

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

Again assuming that the same ontology is used for event literals in FCL and event patterns in BPMN, $\neg \mathit{QualityOfService}$ maps to $m_{ProblemDetected}$ and $\mathit{Replace}$ to $\mathit{FixProblem}$, then P_3 corresponds to a sub-deal situation (see sequence S_4 in Section V-C) for rule $r_{4.2}$, In this case $\mathit{Replace}$ and $\mathit{FixProblem}$ have the same timer. On the other hand, P_4 is not-ideal since the process repairing the violation of the then required event $\mathit{FixProblem3days}$ 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).

VII. 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 the use of 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. Both of these formalisms employ 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 example 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 examples as in [15]. 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 [9] to represent temporal notions and temporalised normative positions and of [4] 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.

Acknowlegements

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

References

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

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

[3] Business process execution language for web services. version 1.1, 5 May 2003.

[4] J. Broersen, F. Dignum, V. Dignum, and J.-J.Ch. Meyer. Designing a deontic logic of deadlines. In *DEON 2004*, LNCS 3065, pp. 43–56. Springer, 2004.

[5] G. Governatori. Representing business contracts in RuleML. *Int. J. of Cooperative Inf. Sys.*, 14(2-3):181–216, 2005.

[6] G. Governatori and D. P. Hoang. A semantic web based architecture for e-contracts in defeasible logic. In *RuleML*, LNCS 3791, pp. 145–159. Springer, 2005.

[7] G. Governatori and Z, Milosevic. Dealing with contract violations: formalism and domain specific language. In *EDOC 2005*, pp. 46–57. IEEE Press, 2005.

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

[9] G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *ICAIL05*, pp. 25–34. ACM Press, 2005.

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

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

[12] R. Lu, S. Sadiq, V. Padmanabhan, and G. Governatori. Using a temporal constraint network for business process execution. In *ADC2006*, CRPIT 49, pp. 157–166, ACS, 2006.

[13] Object Management Group. Business process modeling notation specification, 3 February 2006.

[14] R. Tagg, Z. Milosevic, S. Kulkarni, and S. Gibson. Supporting contract execution through recommended workflows. In *DEXA 2004*, LNCS 3180, pp. 1–12. Springer, 2004.

[15] A.Z. Wyner. A functional program for agents, actions, and deontic specifications. In *Proceedings of DALT 2006*, 2006.