

Dealing with contract violations: formalism and domain specific language

¹Guido Governatori and ²Zoran Milosevic

School of Information Technology and Electrical Engineering
The University of Queensland, Australia
guido@itee.uq.edu.au

Deontik
zoran@deontik.com

December 1, 2005

Overview

- Definition of contract
- Deontic concepts
- Logic of violations
- Logic of violations and BCL

Motivation

Implement contract monitoring:

Motivation

Implement contract monitoring:

Do we know what we have to do according to a contract?

Motivation

Implement contract monitoring:

Do we know what we have to do according to a contract?

- analyse the expected behaviour of the signatories in a precise way;

Motivation

Implement contract monitoring:

Do we know what we have to do according to a contract?

- analyse the expected behaviour of the signatories in a precise way;
- identify and make evident the mutual relationships among various clauses in a contract;

Motivation

Implement contract monitoring:

Do we know what we have to do according to a contract?

- analyse the expected behaviour of the signatories in a precise way;
- identify and make evident the mutual relationships among various clauses in a contract;
- identify the responses to triggering events/conditions as specified in a contract.

What's a contract?

A contract is a declarative act jointly performed also by all parties whose status is going to be changed by the declaration they are performing

What's a contract?

A contract is a declarative act jointly performed also by all parties whose status is going to be changed by the declaration they are performing

Italian Civil Code, art. 1321

A contract is an agreement between two or more parties to create, regulate, or extinguish any legal relationship between them.

Key components of contracts

A contract is a set of clauses

- Definitional clauses
- Prescriptive clauses
 - obligations
 - permissions
 - prohibitions
 - violations

Example

Contract fragment

- 3.1 A “Premium Customer” is a customer who has spent more than \$10000 in goods.
- 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.
- 5.3 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 (Service) is not delivered.

Deontic Logic

Extension of classical logic with the operators OBL_S and PER_S .

- $SpecialOrderPrice(x) = Price(x) + 5\%$
- $OBL_{Supplier} QualityOfService$
- $PER_{Purchaser} ChargeSupplier$

Deontic Logic

Extension of classical logic with the operators OBL_S and PER_S .

- $SpecialOrderPrice(x) = Price(x) + 5\%$
- $OBL_{Supplier} QualityOfService$
- $PER_{Purchaser} ChargeSupplier$

Standard Deontic Logic is not able to deal with violations

Violation paradox

A contract for presentations at EDOC 2005

- Guido should not tell lies in his presentation
- If Guido tells a lie then he has to explain why
- It ought to be the case that if Guido does not tell a lie then he does not explains why
- Guido tells lies in his presentation

Violation paradox

A contract for presentations at EDOC 2005

- Guido should not tell lies in his presentation
 - If Guido tells a lie then he has to explain why
 - It ought to be the case that if Guido does not tell a lie then he does not explains why
 - Guido tells lies in his presentation
-
- $OBL_G \neg lie$
 - $lie \rightarrow OBL_G explain$
 - $OBL_G (\neg lie \rightarrow \neg explain)$
 - lie

Violation paradox

A contract for presentations at EDOC 2005

- Guido should not tell lies in his presentation
- If Guido tells a lie then he has to explain why
- It ought to be the case that if Guido does not tell a lie then he does not explains why
- Guido tells lies in his presentation

- $OBL_G \neg lie$
- $lie \rightarrow OBL_G explain$
- $OBL_G (\neg lie \rightarrow \neg explain)$
- lie

$OBL_G explain$ and $OBL_G \neg explain$

Logic of Violations (FCL)

- 1 A (normative) contract clause is represented by a rule
 $r : A_1, \dots, A_n \vdash \mathbf{X}B$.

Logic of Violations (FCL)

- 1 A (normative) contract clause is represented by a rule
 $r : A_1, \dots, A_n \vdash \mathbf{X}B$.
- 2 A violation does not exist without an obligation it violates.

Logic of Violations (FCL)

- 1 A (normative) contract clause is represented by a rule
 $r : A_1, \dots, A_n \vdash \mathbf{X}B$.
- 2 A violation does not exist without an obligation it violates.
- 3 A reparation of a violation does not exist without a violation it repairs.

Logic of Violations (FCL)

- 1 A (normative) contract clause is represented by a rule
 $r : A_1, \dots, A_n \vdash \mathbf{X}B$.
- 2 A violation does not exist without an obligation it violates.
- 3 A reparation of a violation does not exist without a violation it repairs.
- 4 A reparation can be an obligation itself, and thus it can be violated.

Logic of Violations (FCL)

- 1 A (normative) contract clause is represented by a rule $r : A_1, \dots, A_n \vdash \mathbf{X}B$.
- 2 A violation does not exist without an obligation it violates.
- 3 A reparation of a violation does not exist without a violation it repairs.
- 4 A reparation can be an obligation itself, and thus it can be violated.
- 5 Permissions cannot be violated.

Logic of Violations (FCL)

- ① A (normative) contract clause is represented by a rule
 $r : A_1, \dots, A_n \vdash \mathbf{X}B$.
 - ② A violation does not exist without an obligation it violates.
 - ③ A reparation of a violation does not exist without a violation it repairs.
 - ④ A reparation can be an obligation itself, and thus it can be violated.
 - ⑤ Permissions cannot be violated.
- Contract clauses cannot be taken in isolation.

Logic of Violations (FCL)

- 1 A (normative) contract clause is represented by a rule $r : A_1, \dots, A_n \vdash \mathbf{X}B$.
 - 2 A violation does not exist without an obligation it violates.
 - 3 A reparation of a violation does not exist without a violation it repairs.
 - 4 A reparation can be an obligation itself, and thus it can be violated.
 - 5 Permissions cannot be violated.
- Contract clauses cannot be taken in isolation.
 - It is possible to have chains of obligations/violations

Logic of Violations (FCL)

- ① A (normative) contract clause is represented by a rule
 $r : A_1, \dots, A_n \vdash \mathbf{X}B$.
 - ② A violation does not exist without an obligation it violates.
 - ③ A reparation of a violation does not exist without a violation it repairs.
 - ④ A reparation can be an obligation itself, and thus it can be violated.
 - ⑤ Permissions cannot be violated.
- Contract clauses cannot be taken in isolation.
 - It is possible to have chains of obligations/violations
 - New contract clauses can be derived from the given contract clauses

Making it explicit

- Introducing the reparation operator \otimes

$$EDOC \vdash OBL_G \neg \text{lie} \otimes OBL_G \text{explain} \otimes PER_G \text{apologise}$$

- Merging rules to obtain new rules
- Removing redundancies
- Detecting conflicts

Merging rules

$$\frac{\Gamma \vdash \text{OBL}_S A \quad \Delta, \neg A \vdash \mathbf{X}_{S'} B}{\Gamma, \Delta \vdash \text{OBL}_S A \otimes \mathbf{X}_{S'} B}$$

Merging rules

$$\frac{\Gamma \vdash \text{OBL}_S A \quad \Delta, \neg A \vdash \mathbf{X}_{S'} B}{\Gamma, \Delta \vdash \text{OBL}_S A \otimes \mathbf{X}_{S'} B}$$

Example

From

$$\begin{aligned} & \text{EDOC} \vdash \text{OBL}_G \neg \text{lie} \\ & \text{ChairAuthorisation2Lie}, \text{lie} \vdash \text{OBL}_G \text{explain} \end{aligned}$$

we obtain

$$\text{EDOC}, \text{ChairAuthorisation2Lie} \vdash \text{OBL}_G \neg \text{lie} \otimes \text{OBL}_G \text{explain}$$

What's BCL (Business Contract Language)

BCL is an event driven domain specific language for contract monitoring.

It provides facilities for

- communities
- roles
- policies
- behaviours
- modality (obligations, permissions, prohibitions)
- events and event patters (violations events)
- triggers
- states

BCL policies

BCL basic policy

Policy: QoSPolicy

Role: Supplier

Modality: Obligation

Trigger: SystemStart

Behaviour:

QoS Agreement at <http://supplier/qos1.htm>

Policy:

Role:

Modality:

Trigger:

Behaviour:

EventPattern

BCL Violation policy

Policy: Replace3daysPolicy

Role: Supplier

Modality: Obligation

Guard: HasOccurred QoSPolicy Violation

Behaviour:

Replace.now + 3 days

Mapping notation

$$r_i : \underbrace{A_1^i, \dots, A_n^i}_{Ant(r_i)} \vdash \underbrace{B^i}_{Con(r_i)}$$

where

- each A_j^i is either an event (pattern), or a state (variable)
- B^i is either
 - $X_s C^i$ or
 - $OBL_s C \otimes B^j$

Simple mapping

$$\text{map}(r_i : A_1^i, \dots, A_n^i \vdash X_s C) =$$

Policy: $\text{id} = r_i$

Role: $\text{role}(\text{Con}(r_i))$

Modality: $\text{modality}(\text{Con}(r_i))$

Trigger: $\text{parseEvents}(\text{Ant}(r_i))$

Guard: $\text{parseState}(\text{Ant}(r_i))$

Behaviour: $\text{behaviour}(\text{Con}(r_i))$

Complex mapping

$$\text{map}(r_i : A_1^i, \dots, A_n^i \vdash X_s C \otimes D^i) =$$

Policy: $\text{id}=r_i$

Role: $\text{role}(\text{Con}(r_i))$

Modality: Obligation

Trigger: $\text{parseEvents}(\text{Ant}(r_i))$

Guard: $\text{parseStates}(\text{Ant}(r_i))$

Behaviour: $\text{behaviour}(\text{Con}(r_i))$

$\text{vmap}(D^i, r_i, 0)$

Recursive mapping

$$vmap(D^i, r_i, n) =$$

Policy: $id=r_i.n$

Role: $role(Con(r_i))$

Modality: Obligation

Trigger: SystemStart

Guard: HasOccured r_i Violated

Behaviour: $behaviour(Con(r_i))$

$$vmap(D^i, r_i, n + 1)$$

Example

$$6.1 : Invoice \vdash O_{Purchaser} PayWithin7Days \otimes \\ O_{Purchaser} PayWithInterest.$$

Policy: id=6.1

Role: Purchaser

Modality: Obligation

Trigger: Invoice

Behaviour: PayWithin7Days

$vmap(O_{Purchaser} PayWithInterest, 6.1, 0)$

Example

6.1 : *Invoice* $\vdash O_{Purchaser} PayWithin7Days \otimes$
 $O_{Purchaser} PayWithInterest.$

Policy: id=6.1

Role: Purchaser

Modality: Obligation

Trigger: Invoice

Behaviour: PayWithin7Days

$vmap(O_{Purchaser} PayWithInterest, 6.1, 0)$

Policy: id=6.1.0

Role: Purchaser

Modality: Obligation

Trigger: SystemStart

Guard: HasOccured 6.1 Violated

Behaviour: PayWithInterest

Conclusions

- formal tool to analyse contracts
- mapping to contract monitoring system
- limitations of the two approaches
 - FCL does not cover temporal aspects, limited treatment of events
 - BCL does not cover conflicts, priorities.

Conclusions

- formal tool to analyse contracts
- mapping to contract monitoring system
- limitations of the two approaches
 - FCL does not cover temporal aspects, limited treatment of events
 - BCL does not cover conflicts, priorities.
- In CoALa we show how to deal with conflicts and priorities in FCL.

Conclusions

- formal tool to analyse contracts
- mapping to contract monitoring system
- limitations of the two approaches
 - FCL does not cover temporal aspects, limited treatment of events
 - BCL does not cover conflicts, priorities.
- In CoALa we show how to deal with conflicts and priorities in FCL.
- In CoALa we presented a mapping from BCL to FCL