

# A Semantic Web Based Architecture for e-Contracts in Defeasible Logic

Guido Governatori and Duy Hoang Pham

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

11 November 2005

# Overview

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Motivation

- Definition of contract
- Deontic concepts
- Logic of violations
- Defeasible Deontic Logic of Violations
- Semantic Web architecture for e-contracts

# Motivation

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Motivation

- To provide a vendor neutral Semantic Web based implementation for representing, reasoning and monitoring e-contracts.
- extend RuleML with normative concepts and modalities for e-contracts.
  - The current version of RuleML (0.89) does not support modal operators.

# What's a contract?

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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?

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

A contract is a set of clauses

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

# Example

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

## Contract fragment

- 3.1 A “Premium Customer” is a customer who has spent more than \$10000 in goods.
- 3.2 Services marked as “special order” are subject to a 5% surcharge. Premium customers are exempt from special order surcharge.
- 5.2 The (Supplier) shall on receipt of a purchase order for (Services) make them available within one day.
- 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

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

Extension of classical logic with the operators OBL and PER.

- $SpecialOrderPrice(x) = Price(x) + 5\%$
- $OBL_{Supplier} MakeGoodsAvailble1Day$
- $PER_{Purchaser} ChargeSupplier$

# Deontic Logic

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

Extension of classical logic with the operators OBL and PER.

- $SpecialOrderPrice(x) = Price(x) + 5\%$
- $OBL_{Supplier} MakeGoodsAvailble1Day$
- $PER_{Purchaser} ChargeSupplier$

Standard Deontic Logic is not able to deal with violations

# Violation paradox

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

## A contract for presentations at RuleML 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 explain why
- Guido tells lies in his presentation

# Violation paradox

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

## A contract for presentations at RuleML 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 explain why
  - Guido tells lies in his presentation
- 
- $OBL\neg lie$
  - $lie \rightarrow OBL explain$
  - $OBL(\neg lie \rightarrow \neg explain)$
  - $lie$

# Violation paradox

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

## A contract for presentations at RuleML 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 explain why
- Guido tells lies in his presentation

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

$OBL explain$  and  $OBL\neg explain$

# Logic of Violations

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

# Logic of Violations

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

# Logic of Violations

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract  
Deontic Logic

Logic of  
Violations

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

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract  
Deontic Logic

Logic of  
Violations

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

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

# Making it explicit

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

- Introducing the reparation operator  $\otimes$

$RuleML \vdash OBL\neg lie \otimes OBLexplain \otimes PERapologise$

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

# Making it explicit

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

- Introducing the reparation operator  $\otimes$

$RuleML \vdash OBL\text{-}lie \otimes OBL\textit{explain} \otimes PER\textit{apologise}$

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

See

Guido Governatori and Antonino Rotolo. Logic of Violations. A Gentzen Systems for Reasoning with Contrary-to-Duty Obligations. *Australasian Journal of Logic*, 3, 2005.

Guido Governatori and Zoran Milosevic. An Approach for Validating BCL Contract Specifications. In *Proceedings on the 2nd EDOC Workshop on Contract Architectures and Languages (CoALa 2005)*. IEEE Digital Library.

# Merging rules

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

# Merging rules

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

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

## Example

From

*RuleML*  $\vdash$  *OBL* $\neg$ *lie*

*ChairAuthorisation2Lie*, *lie*  $\vdash$  *OBL**explain*

we obtain

*RuleML*, *ChairAuthorisation2Lie*  $\vdash$  *OBL* $\neg$ *lie*  $\otimes$  *OBL**explain*

# Removing redundancies

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

## Example 1

$$r_1 : RuleML \vdash OBL \neg lie$$

$$r_2 : RuleML \vdash OBL \neg lie \otimes OBL explain$$

# Removing redundancies

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

## Example 1

$$r_1 : RuleML \vdash OBL \neg lie$$

$$r_2 : RuleML \vdash OBL \neg lie \otimes OBL explain$$

## Example 2

$$r_1 : RuleML, lie \vdash OBL explain$$

$$r_2 : RuleML \vdash OBL \neg lie \otimes OBL explain$$

# Removing redundancies

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Contract

Deontic Logic

Logic of  
Violations

## Example 1

$$\begin{aligned}r_1 &: RuleML \vdash OBL\neg lie \\ r_2 &: RuleML \vdash OBL\neg lie \otimes OBLexplain\end{aligned}$$

## Example 2

$$\begin{aligned}r_1 &: RuleML, lie \vdash OBLexplain \\ r_2 &: RuleML \vdash OBL\neg lie \otimes OBLexplain\end{aligned}$$

The normative content of  $r_1$  is included in  $r_2$ , and thus  $r_1$  can be removed

# Why Defeasible Logic/Courteous Logic Programming

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

## Rule-based non-monotonic formalism

- Flexible
- Efficient (linear complexity)
- Directly skeptic semantics
- Argumentation semantics
- Constrictive proof theory
- Encompasses other non-monotonic formalisms used in AI and Law
- Proposed as inferential engine for RuleML
- Applied in several fields/optimised implementations
- Extensible

# Why Defeasible Logic/Courteous Logic Programming

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

## Rule-based non-monotonic formalism

- Flexible
- Efficient (linear complexity)
- Directly skeptic semantics
- Argumentation semantics
- Constructive proof theory
- Encompasses other non-monotonic formalisms used in AI and Law
- Proposed as inferential engine for RuleML
- Applied in several fields/optimised implementations
- Extensible

# Basics of Defeasible Logic

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

- Derive (plausible) conclusions with the minimum amount of information.
  - Definite conclusions
  - Defeasible conclusions
- Defeasible Theory
  - Facts
  - Strict rules ( $A \rightarrow B$ )
  - Defeasible rules ( $A \Rightarrow B$ )
  - Defeaters ( $A \rightsquigarrow B$ )
  - Superiority relation over rules

# Proving Conclusions in Defeasible Logic

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

- 1 Give an argument for the conclusion you want to prove

# Proving Conclusions in Defeasible Logic

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

- 1 Give an argument for the conclusion you want to prove
- 2 Consider all possible counterarguments to it

# Proving Conclusions in Defeasible Logic

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

- 1 Give an argument for the conclusion you want to prove
- 2 Consider all possible counterarguments to it
- 3 Rebut all counterarguments

# Proving Conclusions in Defeasible Logic

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

- 1 Give an argument for the conclusion you want to prove
- 2 Consider all possible counterarguments to it
- 3 Rebut all counterarguments
  - Defeat the argument by a stronger one
  - Undercut the argument by showing that some of the premises do not hold

# Example

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

Facts:  $A_1, A_2, B_1, B_2$

Rules:  $r_1:A_1 \Rightarrow C$

$r_2:A_2 \Rightarrow C$

$r_3:B_1 \Rightarrow \neg C$

$r_4:B_2 \Rightarrow \neg C$

$r_5:B_3 \Rightarrow \neg C$

Superiority relation:

$r_1 > r_3$

$r_2 > r_4$

$r_5 > r_1$

# Example

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

Facts:  $A_1, A_2, B_1, B_2$

Rules:  $r_1:A_1 \Rightarrow C$

$r_2:A_2 \Rightarrow C$

$r_3:B_1 \Rightarrow \neg C$

$r_4:B_2 \Rightarrow \neg C$

$r_5:B_3 \Rightarrow \neg C$

Phase 1: Argument for  $C$

Superiority relation:

$r_1 > r_3$

$r_2 > r_4$

$r_5 > r_1$

# Example

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

Facts:  $A_1, A_2, B_1, B_2$

Rules:  $r_1:A_1 \Rightarrow C$

$r_2:A_2 \Rightarrow C$

$r_3:B_1 \Rightarrow \neg C$

$r_4:B_2 \Rightarrow \neg C$

$r_5:B_3 \Rightarrow \neg C$

Phase 1: Argument for  $C$   
 $A_1$  (Fact),  $r_1 : A_1 \Rightarrow C$

Superiority relation:

$r_1 > r_3$

$r_2 > r_4$

$r_5 > r_1$

# Example

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

Facts:  $A_1, A_2, B_1, B_2$

Rules:  $r_1:A_1 \Rightarrow C$

$r_2:A_2 \Rightarrow C$

$r_3:B_1 \Rightarrow \neg C$

$r_4:B_2 \Rightarrow \neg C$

$r_5:B_3 \Rightarrow \neg C$

Superiority relation:

$r_1 > r_3$

$r_2 > r_4$

$r_5 > r_1$

Phase 1: Argument for  $C$

$A_1$  (Fact),  $r_1 : A_1 \Rightarrow C$

Phase 2: Possible counterarguments

# Example

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

Facts:  $A_1, A_2, B_1, B_2$

Rules:  $r_1: A_1 \Rightarrow C$

$r_2: A_2 \Rightarrow C$

$r_3: B_1 \Rightarrow \neg C$

$r_4: B_2 \Rightarrow \neg C$

$r_5: B_3 \Rightarrow \neg C$

Phase 1: Argument for  $C$

$A_1$  (Fact),  $r_1: A_1 \Rightarrow C$

Phase 2: Possible counterarguments

$r_3: B_1 \Rightarrow \neg C$

$r_4: B_2 \Rightarrow \neg C$

$r_5: B_3 \Rightarrow \neg C$

Superiority relation:

$r_1 > r_3$

$r_2 > r_4$

$r_5 > r_1$

# Example

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

Facts:  $A_1, A_2, B_1, B_2$

Rules:  $r_1:A_1 \Rightarrow C$

$r_2:A_2 \Rightarrow C$

$r_3:B_1 \Rightarrow \neg C$

$r_4:B_2 \Rightarrow \neg C$

$r_5:B_3 \Rightarrow \neg C$

Superiority relation:

$r_1 > r_3$

$r_2 > r_4$

$r_5 > r_1$

Phase 1: Argument for  $C$

$A_1$  (Fact),  $r_1 : A_1 \Rightarrow C$

Phase 2: Possible counterarguments

$r_3 : B_1 \Rightarrow \neg C$

$r_4 : B_2 \Rightarrow \neg C$

$r_5 : B_3 \Rightarrow \neg C$

Phase 3: Rebut the counterarguments

# Example

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

Facts:  $A_1, A_2, B_1, B_2$

Rules:  $r_1:A_1 \Rightarrow C$

$r_2:A_2 \Rightarrow C$

$r_3:B_1 \Rightarrow \neg C$

$r_4:B_2 \Rightarrow \neg C$

$r_5:B_3 \Rightarrow \neg C$

Superiority relation:

$r_1 > r_3$

$r_2 > r_4$

$r_5 > r_1$

Phase 1: Argument for  $C$

$A_1$  (Fact),  $r_1 : A_1 \Rightarrow C$

Phase 2: Possible counterarguments

$r_3 : B_1 \Rightarrow \neg C$

$r_4 : B_2 \Rightarrow \neg C$

$r_5 : B_3 \Rightarrow \neg C$

Phase 3: Rebut the counterarguments

$r_3$  weaker than  $r_1$

$r_4$  weaker than  $r_2$

$r_5$  is not applicable

# Extending the language of Defeasible Logic

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

- extend the language with the deontic operators OBL and PER.
- extend the language with the reparation operator. Permitted only in the head/conclusion of rules.

# Extending the proof mechanism of Defeasible Logic

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

- To prove  $OBLp_n$  from a rule  $A \Rightarrow OBLp_1 \otimes \dots \otimes OBLp_{n-1} \otimes OBLp_n$ , we have to show that  $\neg p_1, \dots, \neg p_{n-1}$  are provable.
- To disprove  $OBLp_n$  from a rule  $A \Rightarrow OBLp_1 \otimes \dots \otimes OBLp_{n-1} \otimes OBLp_n$ , that at least one among  $p_1, \dots, p_{n-1}$  are rejected.

For more details see

Guido Governatori, 2005. Representing Business Contracts in RuleML. *International Journal of Cooperative Information Systems*, 14 (2-3): 181–216.

# DR-CONTRACT basic DTD

## DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

```
<!ELEMENT Atom (Not?,Rel,(Ind|Var)*)>
<!ELEMENT Not (Rel)>
<!ELEMENT Rel (#PCDATA)>
<!ELEMENT Var (#PCDATA)>
<!ELEMENT Ind (#PCDATA)>
<!ELEMENT Fact (Atom)>
<!ELEMENT Imp ((Head,Body)|(Body|Head))>
<!ATTLIST Imp label ID strength #PCDATA>
<!ELEMENT Body (And)>
<!ELEMENT And (Atom|Obligation|Permission)*>
<!ELEMENT Head (Atom|Obligation|Permission|Behaviour)>
<!ELEMENT Behaviour ((Obligation)+,Permission?)>
<!ELEMENT Obligation (Not?,Rel,(Ind|Var)*)>
<!ATTLIST Obligation subject IDREF beneficiary IDREF>
<!ELEMENT Permission (Not?,Rel,(Ind|Var)*)>
<!ATTLIST Permission subject IDREF beneficiary IDREF>
```

# DR-CONTRACT extended DTD

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture

```
<!ELEMENT And (Atom|Obligation|Permission|Violation)*>
<!ELEMENT Violation EMPTY>
<!ATTLIST Violation rule IDREF>
<!ELEMENT Behaviour ((Obligation+,Reparation)|
(Obligation*,Permission?))>
<!ELEMENT Reparation EMPTY>
<!ATTLIST Reparation penalty IDREF>
<!ELEMENT Penalty ((Obligation+,Reparation)|
(Obligation*,Permission?))>
```

The elements given in the extended DTD are provided for convenience, they can be simulated in terms of the elements in the basic DTD.

# DR-Contract Architecture

DR-Contract

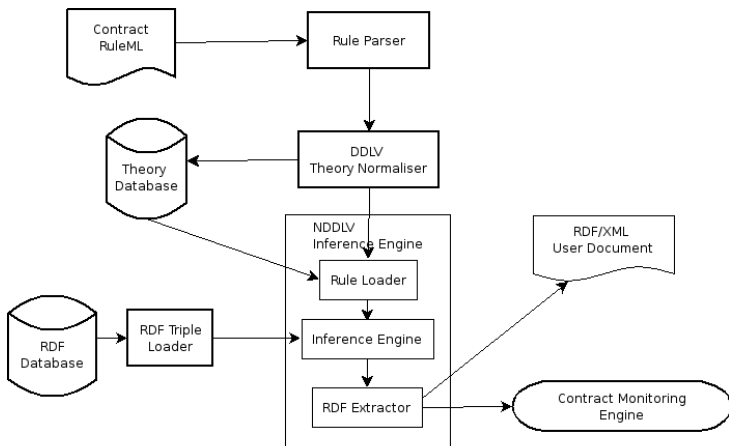
Guido Governatori  
and Duy  
Hoang Pham

Defeasible  
Logic

Extending  
Defeasible  
Logic

Extending  
RuleML

DR-Contract  
Architecture



# Conclusions

DR-Contract

Guido  
Governatori  
and Duy  
Hoang Pham

Our framework to deal with contract

- is grounded on legal theory
- has sound logical foundations
- is backed-up by practice
- is computationally feasible (linear complexity)
- is based on semantic web standards

More work is needed

- optimisation of the implementation
- representation of time
- integration with ontologies
- tools for writing RuleML compliant contracts