Formal Models of Business Process Compliance

Guido Governatori
Marta Indulska
Michael zur Muehlen

JURIX, Rotterdam, 16 December 2009

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Members
NICTA Partners

Overview

- What is compliance
- Deontic concepts
- Logic of violations
- Defeasible Deontic Logic of Violations
- BPM Compliance

Overview

- What is compliance
- Deontic concepts
- Logic of violations
- Defeasible Deontic Logic of Violations
- BPM Compliance
  - Business Process Models

Overview

- What is compliance
- Deontic concepts
- Logic of violations
- Defeasible Deontic Logic of Violations
- BPM Compliance
  - Business Process Models
  - Annotations in execution semantics

Overview

- What is compliance
- Deontic concepts
- Logic of violations
- Defeasible Deontic Logic of Violations
- BPM Compliance
  - Business Process Models
  - Annotations in execution semantics
  - Norms regulating the process

Overview

- What is compliance
- Deontic concepts
- Logic of violations
- Defeasible Deontic Logic of Violations
- BPM Compliance
  - Business Process Models
  - Annotations in execution semantics
  - Norms regulating the process
  - Checking compliance

What's compliance?

Ensuring that business operations, processes, and practices are in accordance with a given prescriptive (often legal) document

What's compliance?

NICTA

Ensuring that business operations, processes, and practices are in accordance with a given prescriptive (often legal) document

Regulatory

- Basel II
- Sarbanes-Oxley
- OFAC (USA Patriot Act)
- OSFI "blocked entity" lists
- HIPAA
- Graham-Leach-Bliley

Standards

- Best practice models
- SAP solution maps
- ISO 9000
- Medical guidelines

Contracts

- Service Agreement
- Customer Contract
- Warranty
- Insurance Policy
- Business Partnership

How to ensure compliance?

How to ensure compliance?

Compliance is a relationship between two sets of specifications

How to ensure compliance?

Compliance is a relationship between two sets of specifications

Alignment of formal specifications for business processes and formal specifications for prescriptive (legal) documents.

Ensuring Compliance

- Ensuring that business processes are in accordance with given prescriptive rules
- Compliance is a relationship between two sets of specifications
- Alignment of formal specifications for business processes and formal specifications for prescriptive rules

Ensuring Compliance

- Ensuring that business processes are in accordance with given prescriptive rules
- Compliance is a relationship between two sets of specifications
- Alignment of formal specifications for business processes and formal specifications for prescriptive rules
- Ensuring that business processes are compliant requires a suitable language for expressing normative specifications in such a way as

Ensuring Compliance

- Ensuring that business processes are in accordance with given prescriptive rules
- Compliance is a relationship between two sets of specifications
- Alignment of formal specifications for business processes and formal specifications for prescriptive rules
- Ensuring that business processes are compliant requires a suitable language for expressing normative specifications in such a way as
  - we can identify formal loopholes, deadlocks and inconsistencies in normative systems, and

Ensuring Compliance

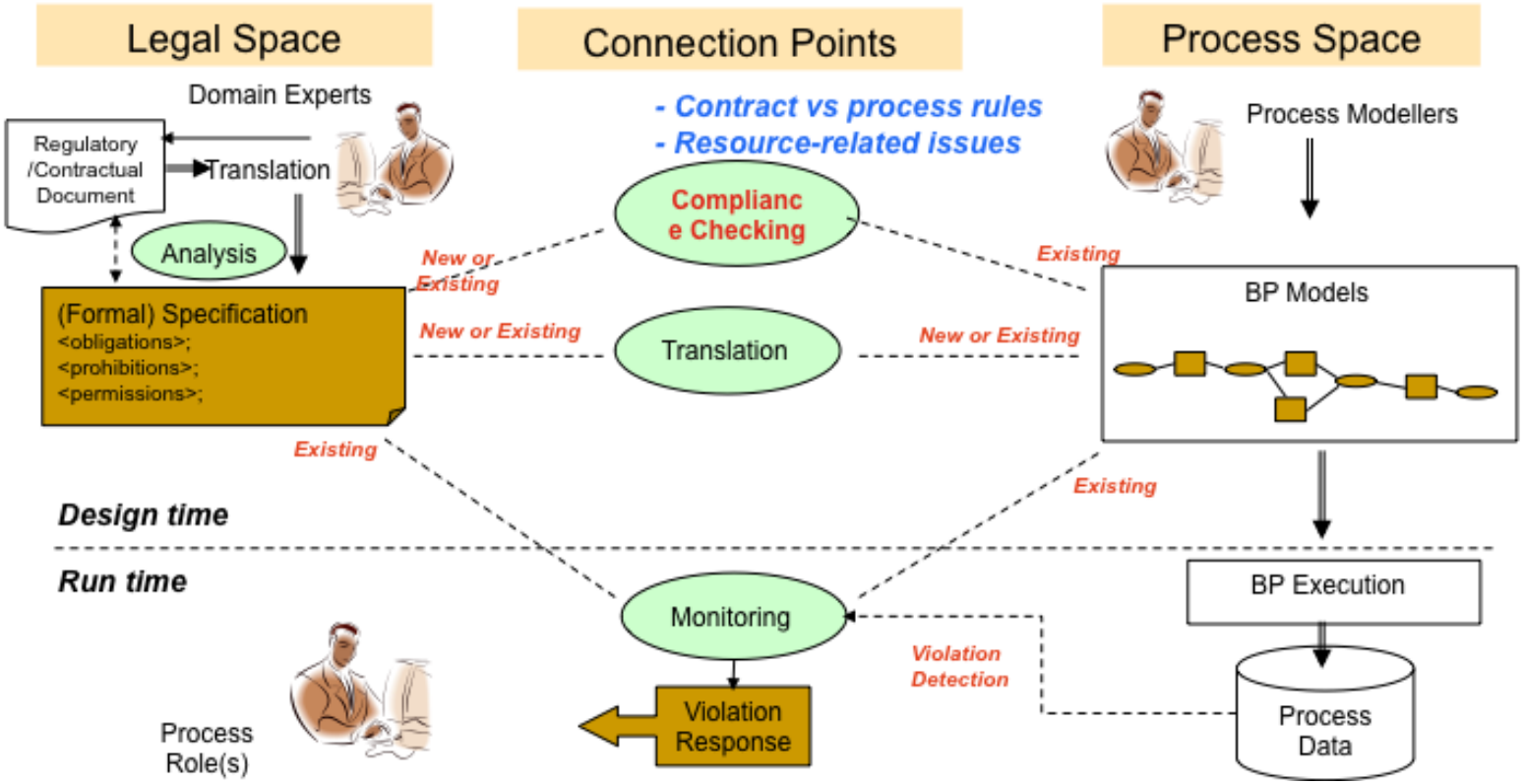
- Ensuring that business processes are in accordance with given prescriptive rules
- Compliance is a relationship between two sets of specifications
- Alignment of formal specifications for business processes and formal specifications for prescriptive rules
- Ensuring that business processes are compliant requires a suitable language for expressing normative specifications in such a way as
  - we can identify formal loopholes, deadlocks and inconsistencies in normative systems, and
  - we can make hidden conditions explicit

Ensuring Compliance

- Ensuring that business processes are in accordance with given prescriptive rules
- Compliance is a relationship between two sets of specifications
- Alignment of formal specifications for business processes and formal specifications for prescriptive rules
- Ensuring that business processes are compliant requires a suitable language for expressing normative specifications in such a way as
  - we can identify formal loopholes, deadlocks and inconsistencies in normative systems, and
  - we can make hidden conditions explicit

Without this, we do not have any guarantee that a given business process is compliant, because we do not know if all relevant norms have been considered

Compliance Ecosystem



Compliance Pipeline

- Internal (Governance) or External (Regulations/Standards/Contracts) generate compliance requirements

Compliance Pipeline

- Internal (Governance) or External (Regulations/Standards/Contracts) generate compliance requirements
- Consultants define *Control Objectives* to fulfill compliance obligations

Compliance Pipeline

- Internal (Governance) or External (Regulations/Standards/Contracts) generate compliance requirements
- Consultants define *Control Objectives* to fulfill compliance obligations
- Control Objectives analysed in terms of *risk* and necessary *internal controls* required for "effective and efficient" implementation of control objectives

Compliance Pipeline

- Internal (Governance) or External (Regulations/Standards/Contracts) generate compliance requirements
- Consultants define *Control Objectives* to fulfill compliance obligations
- Control Objectives analysed in terms of *risk* and necessary *internal controls* required for "effective and efficient" implementation of control objectives
  - Proposed internal controls are built into business processes / transactions (Compliance by design)

Compliance Pipeline

- Internal (Governance) or External (Regulations/Standards/Contracts) generate compliance requirements
- Consultants define *Control Objectives* to fulfill compliance obligations
- Control Objectives analysed in terms of *risk* and necessary *internal controls* required for "effective and efficient" implementation of control objectives
  - Proposed internal controls are built into business processes / transactions (Compliance by design)
  - Operational practices are monitored for deviation from prescribed internal controls (Detection of non-compliance)

Compliance by design methodology

NICTA

Controls Directory Management

Ontological Alignment mapping between the languages, common ontology.

Modeling Control Objectives Formal language based on deontic logic to represent and analyze normative positions stemming from compliance requirements

Example

The creation and approval of purchase requests must be undertaken by two separate purchase officers

$$c1 : CreatePR(x, y), PurchaseOfficer(y) \Rightarrow O\neg ApprovedPR(x, y)$$

Process Model Enrichment attach semantic annotations to activities in a process model (Flow Tags, Data Tags, Resource Tags, Time Tags)

Event Monitoring Operational practices are monitored for deviation from prescribed internal controls

Key components of Normative Systems

A normative system is a set of clauses

- Definitional clauses (counts-as rules)
- Prescriptive clauses (norms)
  - obligations
  - permissions
  - prohibitions
  - violations

Example

NICTA

Contract fragment

3.1 A "Premium Customer" is a customer who has spent more that $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.

Formalising Compliance

- Clausal forms (Sergot-Kowalski, Ghose-Koliadis, Hoffman-Weber-Governatori)

Formalising Compliance

- Clausal forms (Sergot-Kowalski, Ghose-Koliadis, Hoffman-Weber-Governatori)
- Temporal logic (van der Aalst, Rinderle-Ma)

Formalising Compliance

- Clausal forms (Sergot-Kowalski, Ghose-Koliadis, Hoffman-Weber-Governatori)
- Temporal logic (van der Aalst, Rinderle-Ma)
- Event Calculus (Goedertier, Singh, Farrell-Sergot)

Formalising Compliance

- Clausal forms (Sergot-Kowalski, Ghose-Koliadis, Hoffman-Weber-Governatori)
- Temporal logic (van der Aalst, Rinderle-Ma)
- Event Calculus (Goedertier, Singh, Farrell-Sergot)
- Deontic Logic (Governatori-Milosevic-Sadiq, Goedertier, Liu)

Clausal Approaches

$$p \vee (q \wedge r) \vee s$$

- do not distinguish different normative positions

Clausal Approaches

$$p \vee (q \wedge r) \vee s$$

- do not distinguish different normative positions
- directionality of norms (what are the premises and what is the conclusion)

Clausal Approaches

$$p \vee (q \wedge r) \vee s$$

- do not distinguish different normative positions
- directionality of norms (what are the premises and what is the conclusion)
- semantic compliance (annotations)

Temporal Logic

$$p \rightarrow A\Box q$$

and model checking

- Temporal logic and model checking have been used to verification of software and hardware system

Temporal Logic

$$p \rightarrow A\Box q$$

and model checking

- Temporal logic and model checking have been used to verification of software and hardware system
- Mature technology

Temporal Logic

$$p \rightarrow A\Box q$$

and model checking

- Temporal logic and model checking have been used to verification of software and hardware system
- Mature technology
- Structural Compliance only

Temporal Logic

$$p \rightarrow A\Box q$$

and model checking

- Temporal logic and model checking have been used to verification of software and hardware system
- Mature technology
- Structural Compliance only
- Does not distinguish normative positions

Temporal Logic

$$p \rightarrow A\Box q$$

and model checking

- Temporal logic and model checking have been used to verification of software and hardware system
- Mature technology
- Structural Compliance only
- Does not distinguish normative positions
- Standard Deontic Logic can be simulated in Temporal Logic

Temporal Logic

$$p \rightarrow A\Box q$$

and model checking

- Temporal logic and model checking have been used to verification of software and hardware system
- Mature technology
- Structural Compliance only
- Does not distinguish normative positions
- Standard Deontic Logic can be simulated in Temporal Logic
- Permissions must always be instantiated

Deontic Logic

Extension of logic with the operators OBL and PER.

- $SpecialOrderPrice(x) = Price(x) + 5\%$
- $\mathrm{OBL}_{Supplier}\, MakeGoodsAvailble1Day$
- $\mathrm{PER}_{Purchaser}\, ChargeSupplier$

Standard Deontic Logic

Extension of classical logic with the modal operators OBL and PER.

- $\mathrm{OBL}\alpha \equiv \neg\mathrm{PER}\neg\alpha, \quad \mathrm{PER}\alpha \equiv \neg\mathrm{OBL}\neg\alpha$
- $\mathrm{OBL}(\alpha \to \beta) \to (\mathrm{OBL}\alpha \to \mathrm{OBL}\beta)$
- $\mathrm{OBL}\alpha \to \mathrm{PER}\alpha$ or $(\mathrm{OBL}\alpha \to \neg\mathrm{OBL}\neg\alpha)$

Standard Deontic Logic

Extension of classical logic with the modal operators OBL and PER.

- $\mathrm{OBL}\alpha \equiv \neg\mathrm{PER}\neg\alpha, \quad \mathrm{PER}\alpha \equiv \neg\mathrm{OBL}\neg\alpha$
- $\mathrm{OBL}(\alpha \to \beta) \to (\mathrm{OBL}\alpha \to \mathrm{OBL}\beta)$
- $\mathrm{OBL}\alpha \to \mathrm{PER}\alpha$ or $(\mathrm{OBL}\alpha \to \neg\mathrm{OBL}\neg\alpha)$

Standard Deontic Logic is not able to deal with violations

Violation paradox

Rules for JURIX Tutorials

- 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

Rules for JURIX Tutorials

- 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

- $\mathrm{OBL}\neg \mathit{lie}$
- $\mathit{lie} \rightarrow \mathrm{OBL}\mathit{explain}$
- $\mathrm{OBL}(\neg \mathit{lie} \rightarrow \neg \mathit{explain})$
- lie

Violation paradox

NICTA

Rules for JURIX Tutorials

- 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

- $\mathrm{OBL}\neg lie$
- $lie \rightarrow \mathrm{OBL}explain$
- $\mathrm{OBL}(\neg lie \rightarrow \neg explain)$
- lie

$\mathrm{OBL}explain$ and $\mathrm{OBL}\neg explain$

Kobayashi Maru

What's the problem?

What's the problem?

$$a \Rightarrow Ob$$

$$\neg b \Rightarrow Oc$$

$$\neg c \Rightarrow Od$$

What's the problem?

$$a \Rightarrow Ob$$

$$\neg b \Rightarrow Oc$$

$$\neg c \Rightarrow Od$$

What about a, $\neg b$ but d?

What's the problem?

$$a \Rightarrow Ob$$

$$\neg b \Rightarrow Oc$$

$$\neg c \Rightarrow Od$$

What about a, $\neg b$ but d?

A logic of violations

Logic of Violations

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

Logic of Violations

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

Logic of Violations

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

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

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

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

- Prescriptive clauses cannot be taken in isolation.

Logic of Violations

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

- Prescriptive clauses cannot be taken in isolation.
- It is possible to have chains of obligations/violations

Logic of Violations

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

- Prescriptive clauses cannot be taken in isolation.
- It is possible to have chains of obligations/violations
- New prescriptive clauses can be derived from the given prescriptive clauses

Modelling Norms

Norms are modelled as rules in FCL.

Language

- literals $p, q, \ldots$ (atomic proposition and their negation)
- deontic literals Op (Obligatory p), P (Permitted p), Fp (Forbidden p, i.e., $O\neg p$.)

Rules

- Normal rules

$$A_1, \ldots, A_n \Rightarrow OB$$

$A_1 \ldots, A_n$ trigger the obligation of B.

- Rules for violations

$$A_1, \ldots, A_n \Rightarrow OB_1 \otimes OB_2 \otimes OB_3 \otimes \cdots \otimes OB_n$$

$A_1 \ldots, A_n$ trigger the obligation of B_1 but if B_1 is violated then B_2 is obligatory and so on.

Making it explicit

Making it explicit

- Introducing the reparation operator $\otimes$

$$PrimeMinister \Rightarrow O\neg\mathrm{lie} \otimes O\,explain \otimes O\,impeachment$$

Making it explicit

- Introducing the reparation operator $\otimes$

$$PrimeMinister \Rightarrow O\neg\mathrm{lie} \otimes O\,explain \otimes O\,impeachment$$

- Merging rules to obtain new rules

Making it explicit

- Introducing the reparation operator $\otimes$

$$PrimeMinister \Rightarrow O\neg\mathrm{lie} \otimes O\,explain \otimes O\,impeachment$$

- Merging rules to obtain new rules
- Removing redundancies

Making it explicit

- Introducing the reparation operator $\otimes$

$$PrimeMinister \Rightarrow O\neg\mathrm{lie} \otimes O\,explain \otimes O\,impeachment$$

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

Merging rules

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

Merging rules

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

Example

From

$$JURIX \vdash \mathrm{OBL}_{guido} \neg lie$$
$$MartaGrantedLie, lie \vdash \mathrm{OBL}explain$$

we obtain

$$JURIX, MartaGrantedLie \vdash \mathrm{OBL}_{guido} \neg lie \otimes \mathrm{OBL}_{guido} explain$$

Removing redundancies

Example 1

$$r_1 : JURIX \vdash \mathrm{OBL}\neg lie$$

$$r_2 : JURIX \vdash \mathrm{OBL}\neg lie \otimes \mathrm{OBL}explain$$

Removing redundancies

NICTA

Example 1

$$r_1 : JURIX \vdash \mathrm{OBL}\neg lie$$

$$r_2 : JURIX \vdash \mathrm{OBL}\neg lie \otimes \mathrm{OBL}explain$$

Example 2

$$r_1 : JURIX, lie \vdash \mathrm{OBL}explain$$

$$r_2 : JURIX \vdash \mathrm{OBL}\neg lie \otimes \mathrm{OBL}explain$$

Removing redundancies

NICTA

Example 1

$$r_1 : JURIX \vdash \mathrm{OBL}\neg lie$$
$$r_2 : JURIX \vdash \mathrm{OBL}\neg lie \otimes \mathrm{OBL}explain$$

Example 2

$$r_1 : JURIX, lie \vdash \mathrm{OBL}explain$$
$$r_2 : JURIX \vdash \mathrm{OBL}\neg lie \otimes \mathrm{OBL}explain$$

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

Detecting conflicts

$$\frac{\Gamma \vdash A \qquad \Delta \vdash \neg A}{\Gamma, \Delta \vdash \mathit{false}}$$

if Γ and Δ are mutually consistent and A and $\neg A$ cannot be repaired

Detecting conflicts

$$\frac{\Gamma \vdash A \qquad \Delta \vdash \neg A}{\Gamma, \Delta \vdash \mathit{false}}$$

if Γ and Δ are mutually consistent and A and $\neg A$ cannot be repaired

Example

From

$$\mathit{JURIX} \vdash \mathrm{OBL}\mathit{lie}$$

$$\mathit{LongSeminar} \vdash \mathrm{OBL}\neg\mathit{lie}$$

we obtain

$$\mathit{JURIX}, \mathit{LongSeminar} \vdash \mathit{false}$$

Normal FCL Forms[1]

- A 'cleaned-up' version of the FCL specifications
- Related clauses are merged, in particular linking original clauses and reparation clauses
- Removing redundancies, in particular clauses that are subsumed by other clauses
- Detecting and resolving conflicts
- NFCL form is used for compliance checking. NFCL forms describe behavioural and state space of contract

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

Why Non-monotonic Reasoning

Reasonable Results with Minimum Effort

Why Non-monotonic Reasoning

Reasonable Results with Minimum Effort

$$PhD \rightarrow Uni$$

Why Non-monotonic Reasoning

Reasonable Results with Minimum Effort

$$PhD \rightarrow Uni$$
$$Weekend \rightarrow \neg Uni$$
$$PublicHoliday \rightarrow \neg Uni$$
$$Sick \rightarrow \neg Uni$$

Why Non-monotonic Reasoning

Reasonable Results with Minimum Effort

$$
\begin{aligned}
PhD &\rightarrow Uni \\
Weekend &\rightarrow \neg Uni \\
PublicHoliday &\rightarrow \neg Uni \\
Sick &\rightarrow \neg Uni \\
Weekend \wedge VICdeadline &\rightarrow Uni
\end{aligned}
$$

Why Non-monotonic Reasoning

Reasonable Results with Minimum Effort

$$
\begin{aligned}
PhD &\rightarrow Uni \\
Weekend &\rightarrow \neg Uni \\
PublicHoliday &\rightarrow \neg Uni \\
Sick &\rightarrow \neg Uni \\
Weekend \wedge VICdeadline &\rightarrow Uni
\end{aligned}
$$

VIC= Very Important Conference (e.g., Jurix, ICAIL,. . .)

Why Non-monotonic Reasoning

Reasonable Results with Minimum Effort

$$
\begin{aligned}
PhD &\rightarrow Uni \\
Weekend &\rightarrow \neg Uni \\
PublicHoliday &\rightarrow \neg Uni \\
Sick &\rightarrow \neg Uni \\
Weekend \wedge VICdeadline &\rightarrow Uni \\
VICdeadline \wedge PartnerBirthday &\rightarrow \neg Uni
\end{aligned}
$$

VIC= Very Important Conference (e.g., Jurix, ICAIL,. . .)

Why Non-monotonic Reasoning

Reasonable Results with Minimum Effort

$$
\begin{aligned}
PhD &\rightarrow Uni \\
Weekend &\rightarrow \neg Uni \\
PublicHoliday &\rightarrow \neg Uni \\
Sick &\rightarrow \neg Uni \\
Weekend \wedge VICdeadline &\rightarrow Uni \\
VICdeadline \wedge PartnerBirthday &\rightarrow \neg Uni
\end{aligned}
$$

$$Phd \wedge (\neg Weekend \vee (Weekend \wedge VICdeadline \wedge \neg PartnerBirthday)) \wedge \neg Sick \ldots \rightarrow Uni$$

VIC= Very Important Conference (e.g., Jurix, ICAIL,...)

Why Defeasible Logic

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
- Applied in several fields/optimised implementations
- Extensible

Why Defeasible Logic

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
- Applied in several fields/optimised implementations
- Extensible

Basics of Defeasible Logic

- 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

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

Proving Conclusions in Defeasible Logic

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

Proving Conclusions in Defeasible Logic

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

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

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

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

Example

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), $r_1 : A_1 \Rightarrow C$

Example

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

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
$r_3 : B_1 \Rightarrow \neg C$
$r_4 : B_2 \Rightarrow \neg C$
$r_5 : B_3 \Rightarrow \neg C$

Example

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

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

- 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

- To prove $\mathrm{OBL}p_n$ from a rule $A \Rightarrow \mathrm{OBL}p_1 \otimes \cdots \otimes \mathrm{OBL}p_{n-1} \otimes \mathrm{OBL}p_n$, we have to show that $\neg p_1, \ldots, \neg p_{n-1}$ are provable.
- To disprove $\mathrm{OBL}p_n$ from a rule $A \Rightarrow \mathrm{OBL}p_1 \otimes \cdots \otimes \mathrm{OBL}p_{n-1} \otimes \mathrm{OBL}p_n$, that at least one among $p_1, \ldots, p_{n_1}$ 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.

Part I

BPM Compliance

Business Process Model (BPM)

Business Process Model (BPM)

- A BPM describes the tasks to be executed (and the order in which they are executed) to fulfill some objectives

Business Process Model (BPM)

- A BPM describes the tasks to be executed (and the order in which they are executed) to fulfill some objectives
- A language for BPM usually has two elements:

Business Process Model (BPM)

- A BPM describes the tasks to be executed (and the order in which they are executed) to fulfill some objectives
- A language for BPM usually has two elements:
  - Tasks are activities to be performed

Business Process Model (BPM)

- A BPM describes the tasks to be executed (and the order in which they are executed) to fulfill some objectives
- A language for BPM usually has two elements:
  - Tasks are activities to be performed
  - Connectors consist of

Business Process Model (BPM)

- A BPM describes the tasks to be executed (and the order in which they are executed) to fulfill some objectives
- A language for BPM usually has two elements:
  - Tasks are activities to be performed
  - Connectors consist of
    - sequence (a task is performed after another task),

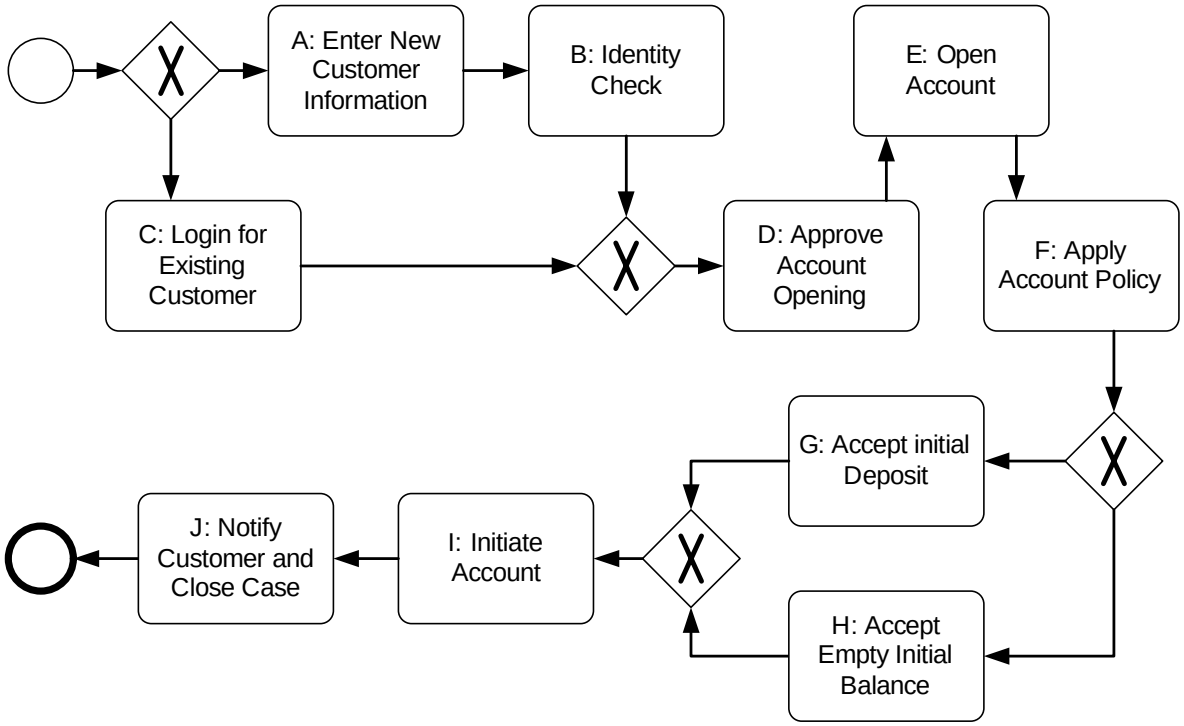
Business Process Model (BPM)

- A BPM describes the tasks to be executed (and the order in which they are executed) to fulfill some objectives
- A language for BPM usually has two elements:
  - Tasks are activities to be performed
  - Connectors consist of
    - sequence (a task is performed after another task),
    - parallel—and-split and and-join—(tasks are to be executed in parallel),

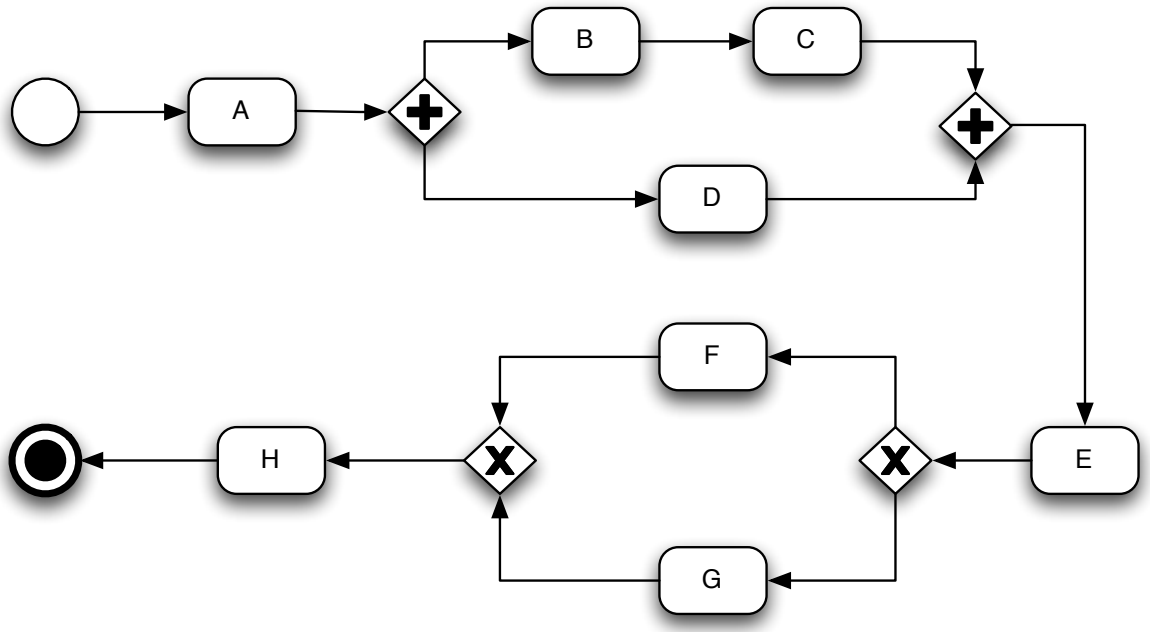
Business Process Model (BPM)

- A BPM describes the tasks to be executed (and the order in which they are executed) to fulfill some objectives
- A language for BPM usually has two elements:
  - Tasks are activities to be performed
  - Connectors consist of
    - sequence (a task is performed after another task),
    - parallel—and-split and and-join—(tasks are to be executed in parallel),
    - choice—(x)or-split and (x)or-join—(at least (most) one task in a set of task must be executed).

Example: Account Opening Process

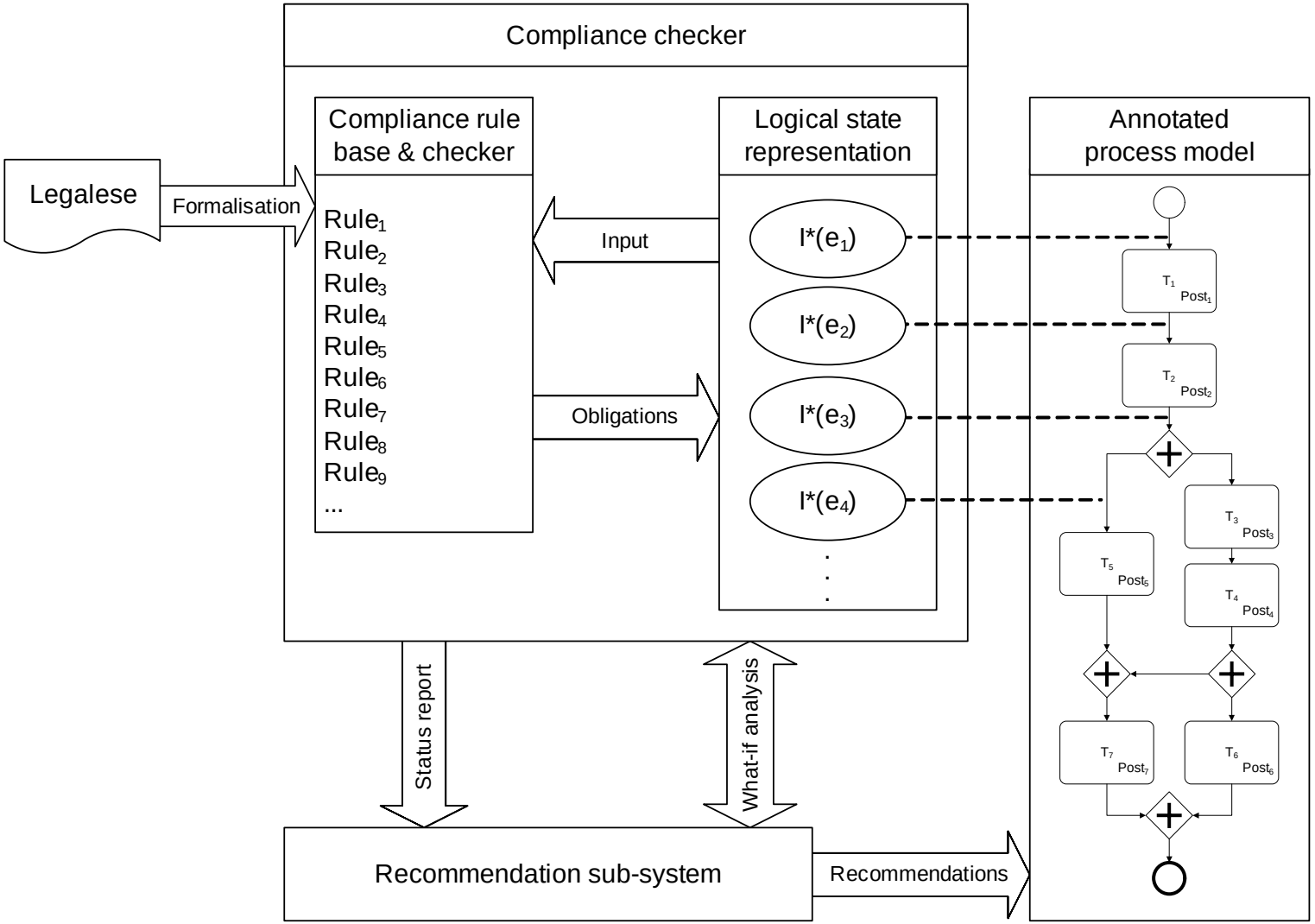


Execution Traces



A, B, C, D, E, F, H
A, D, B, C, E, G, H
A, D, B, C, E, F, H

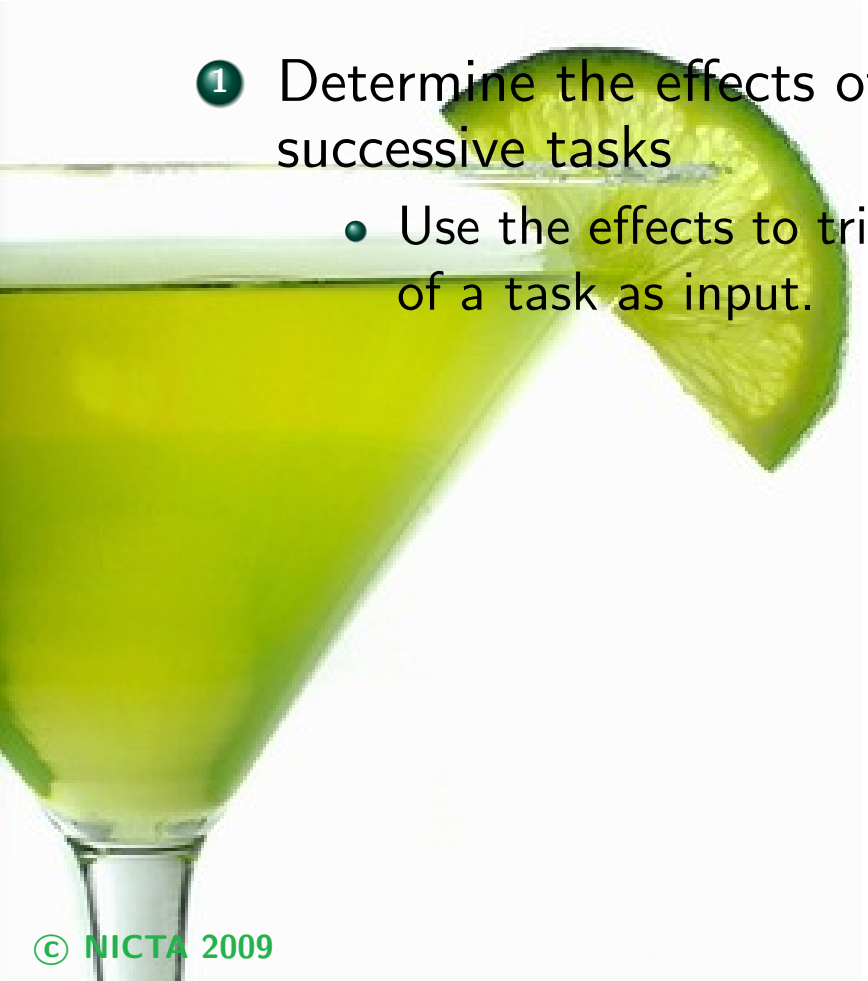
Business Process Compliance



Checking Compliance Recipe

1. Determine the effects of each task and propagate them to successive tasks

Checking Compliance Recipe

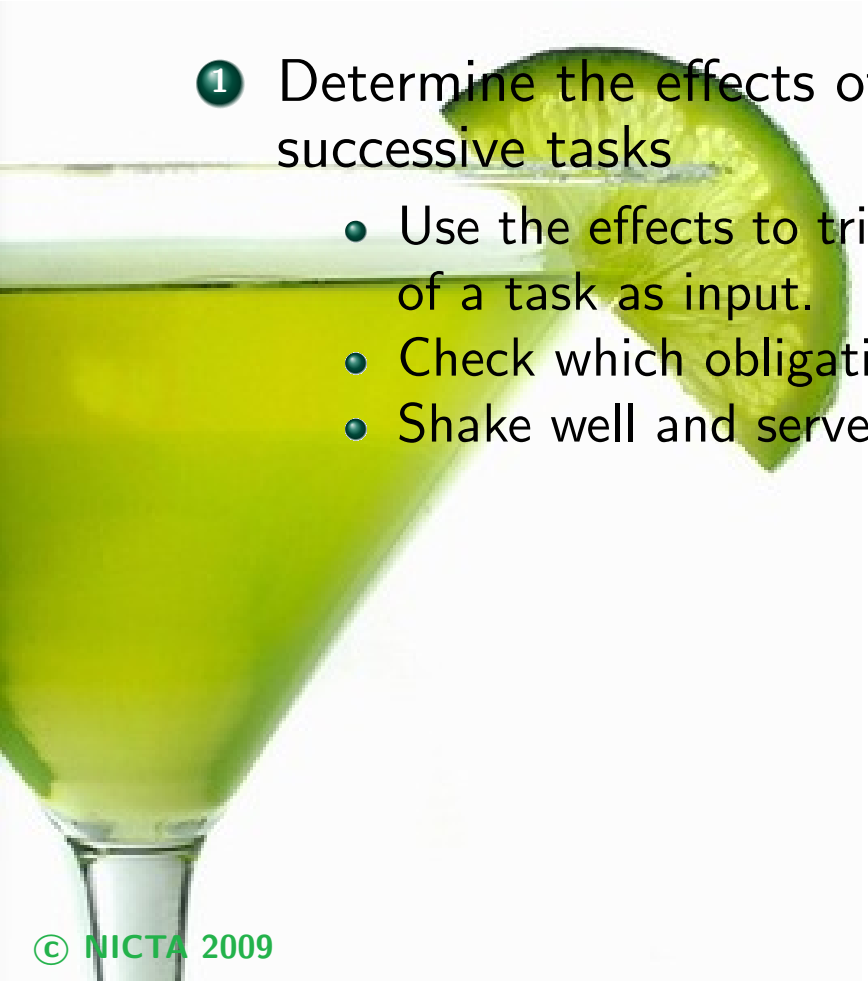


1. Determine the effects of each task and propagate them to successive tasks
   - Use the effects to trigger obligations. Run FCL with the effects of a task as input.

Checking Compliance Recipe

1. Determine the effects of each task and propagate them to successive tasks
   - Use the effects to trigger obligations. Run FCL with the effects of a task as input.
   - Check which obligations have been fulfilled, violated

Checking Compliance Recipe

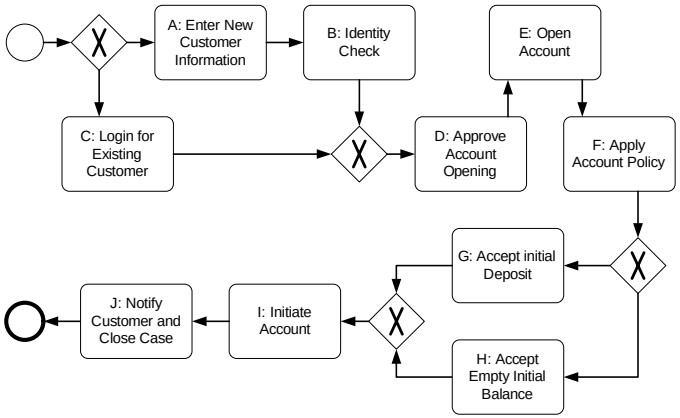


1. Determine the effects of each task and propagate them to successive tasks
   - Use the effects to trigger obligations. Run FCL with the effects of a task as input.
   - Check which obligations have been fulfilled, violated
   - Shake well and serve!

The Journey to Compliance

1. Take or design a business process
2. Annotate the process
   - effects of the tasks (each task is annotated with the effects it produces)
   - rules encoding the norms relevant to the process

Adding Annotations



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

Rules for the Process

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

$$r_1 : newCustomer(x) \Rightarrow O checkIdentity(x)$$

- Retain history of identity checks performed.

$$r_2 : checkIdentity(x) \Rightarrow O recordIdentity(x)$$

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

$$r_3 : account(x) \Rightarrow O positiveBalance(x) \otimes O approveManager(x)$$

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

Ideal Semantics: compliance checking

NICTA

- FCL constraints determine behavioural paths (generic)
  - behavioural paths special case business processes
  - currently expressed as event sequences
- Ideal situation
  - Execution traces do not violate NFCL
- Sub-ideal situation
  - There are violations, but they are repaired/compensated
- Non-ideal (non-compliant) situation
  - There are violations, but they are NOT repaired/compensated
- Irrelevant situation
  - No rule is applicable

Handling Obligations

An obligation chain $OA_1 \otimes \cdots \otimes OA_n$ is *active* given a set of literals S, if

- there is a rule $\Gamma \Rightarrow OA_1 \otimes \cdots \otimes OA_n$ such that $\Gamma \subseteq S$, i.e., the rule is triggered by the situation, and
- for all rule for conflicting chains, either
  - the chain is not triggered by the situation or
  - the negation of an element before the conflicting element is not in the situation.

The Algorithm

Input: *Current* set of active obligation chains

$A_1 \otimes A_2 = C \in Current$

For each $C \in Current$

if $A_1 = OB$, then

  if $B \in S$, then

    remove($[T, R, A_1 \otimes A_2]$, *Current*),

    remove($[T, R, A_1 \otimes A_2]$, *Unfulfilled*)

    if $[T, R, B_1 \otimes B_2 \otimes A_1 \otimes A_2] \in Violated$ then

      add($[T, R, B_1 \otimes B_2 \otimes A_1 \otimes A_2]$, *Compensated*)

  if $\neg B \in S$, then

    add($[T, R, A_1 \otimes A_2]$, *Violated*), add($[T, R, A_2]$, *Current*)

  else

    add($[T, R, A_1 \otimes A_2]$, *Unfulfilled*).

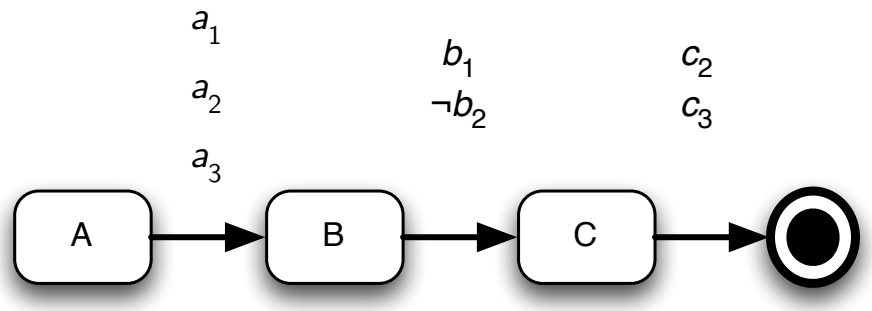
Finally Compliant!

NICTA

Definition

- An execution trace is *compliant* iff for all $[T, R, A] \in \mathit{Current}$, $A = \mathit{OB} \otimes C$, for every $[T, R, A, B] \in \mathit{Violated}$, $[T, R, A, B] \in \mathit{Compensated}$ and $\mathit{Unfulfilled} = \emptyset$.
- An execution trace is *fully compliant* iff for all $[T, R, A] \in \mathit{Current}$, $A = \mathit{OB} \otimes C$, $\mathit{Violated} = \emptyset$ and $\mathit{Unfulfilled} = \emptyset$.
- A process is (*fully*) *compliant* iff all its execution traces are (fully) compliant.

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

Compute the current chains of obligations. Add them to Current

Current

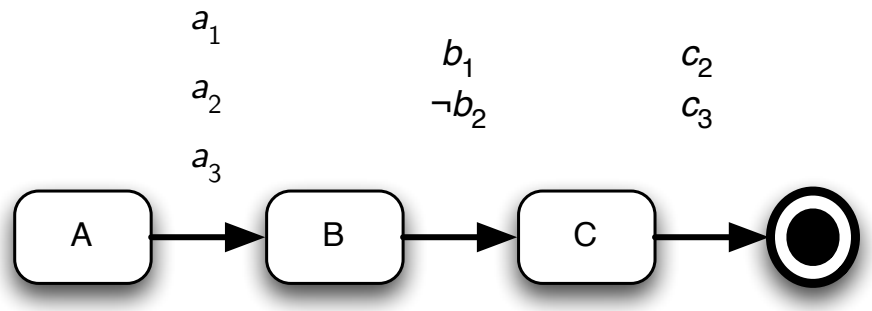
$[A,\ r_1,\ Ob_1 \otimes Oc_1]$
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$

Violated

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

Examine the chain one by one against the effects.

Current

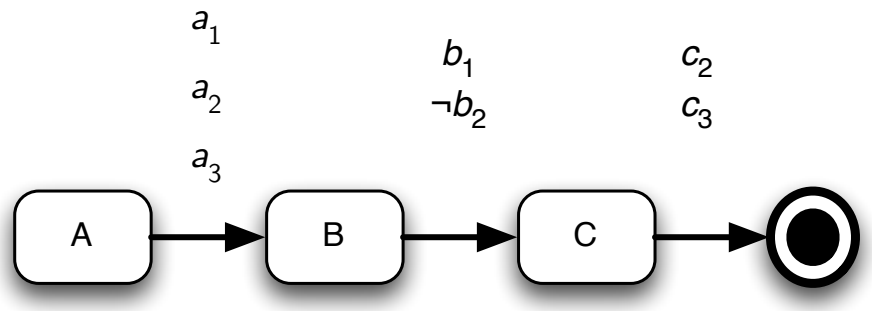
$[A,\ r_1,\ Ob_1 \otimes Oc_1]$
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$

Violated

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

$Ob_1 \otimes Oc_1$ and b_1. Remove the chain from Current

Current

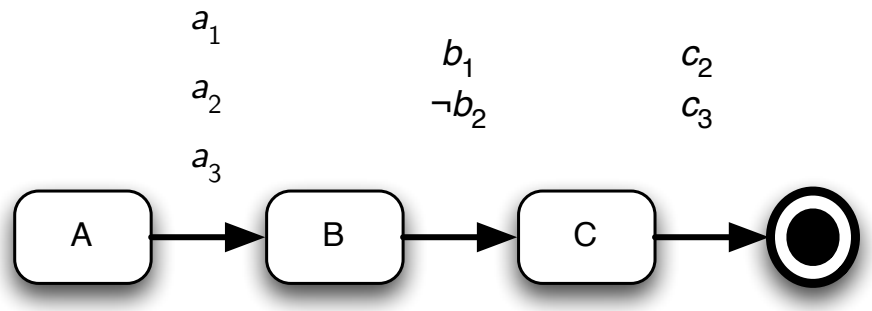
$[A,\ r_1,\ Ob_1 \otimes Oc_1]$
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$

Violated

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

$Ob_1 \otimes Oc_1$ and b_1. Remove the chain from Current

Current

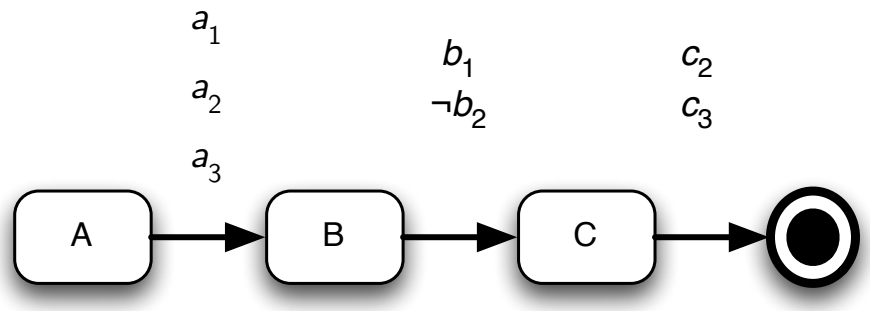
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$

Violated

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

$Ob_2 \otimes Oc_2$ and $\neg b_2$. Violation

Current

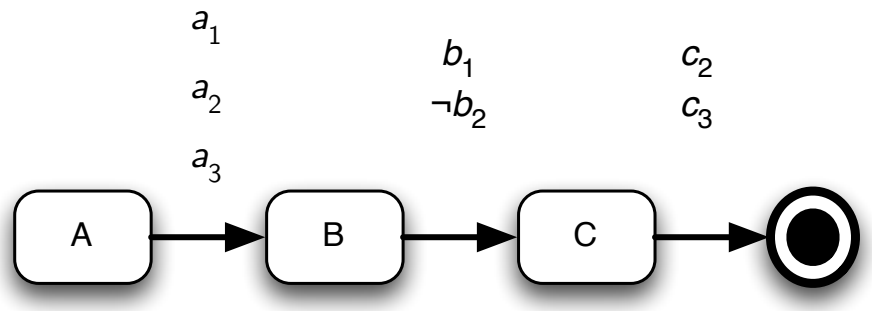
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$

Violated

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

Add the chain to Violated, add the compensatory part to Current

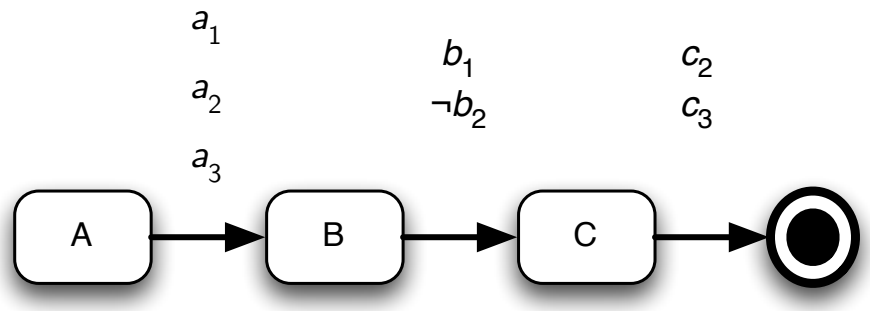
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[B,\ r_2,\ Oc_2]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

$Ob_3 \otimes Oc_3 \otimes Od_3$. No information about b_3

Current

$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[B,\ r_2,\ Oc_2]$

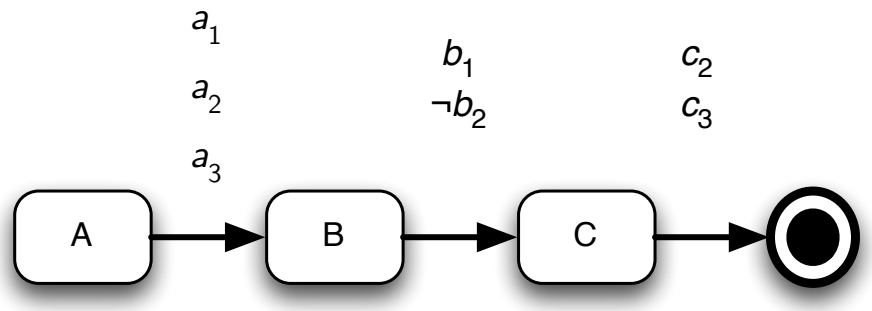
Violated

$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

Remove $Ob_3 \otimes Oc_3 \otimes Od_3$ from Current, add it to Unfulfilled

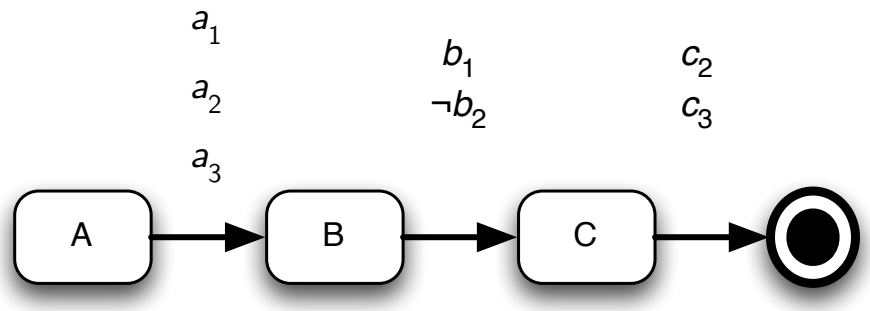
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[B,\ r_2,\ Oc_2]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

Oc_2. No information about c_2

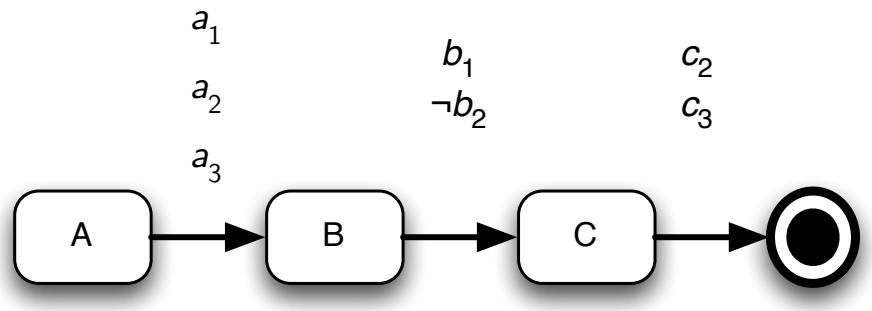
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[B,\ r_2,\ Oc_2]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

Remove Oc_2 from Current, add it to Unfulfilled

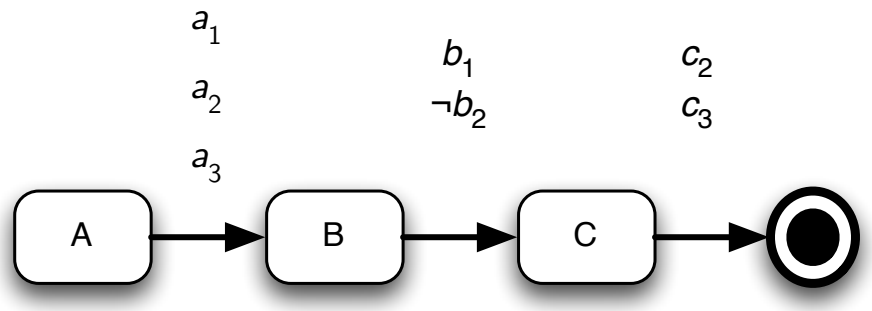
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[B,\ r_2,\ Oc_2]$

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

We have done with B and we move to C

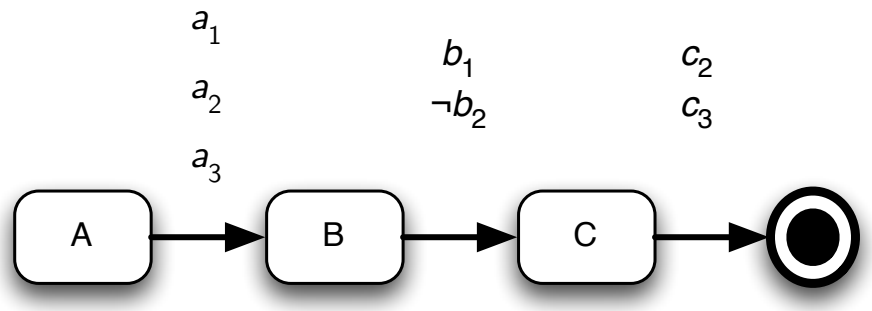
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[B,\ r_2,\ Oc_2]$

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

We move the elements of Unfulfilled to Current

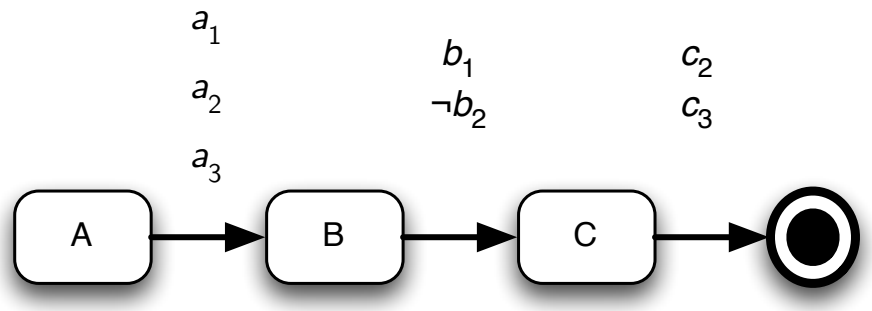
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[B,\ r_2,\ Oc_2]$

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

We move the elements of Unfulfilled to Current

Current

$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[B,\ r_2,\ Oc_2]$

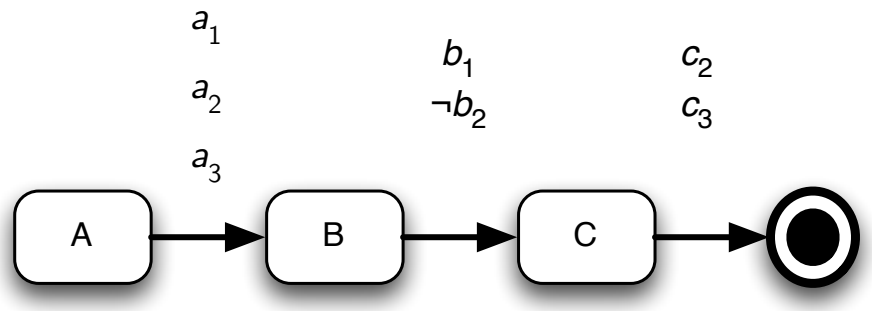
Violated

$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

We compute the set of chains for C

Current

$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[B,\ r_2,\ Oc_2]$
$[C,\ r_5,\ Oc_3]$

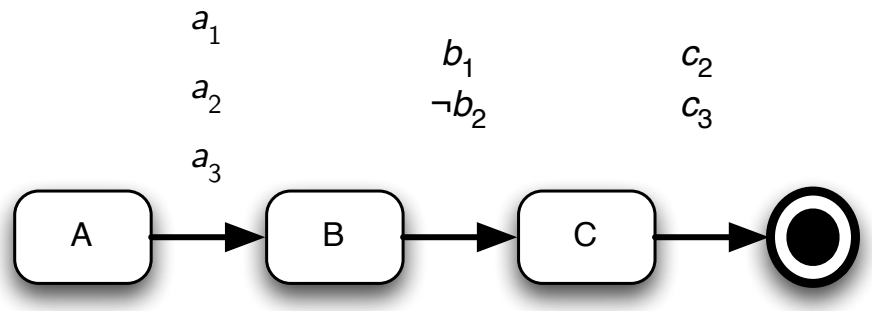
Violated

$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

Oc_2 and c_2, remove the chain from Current

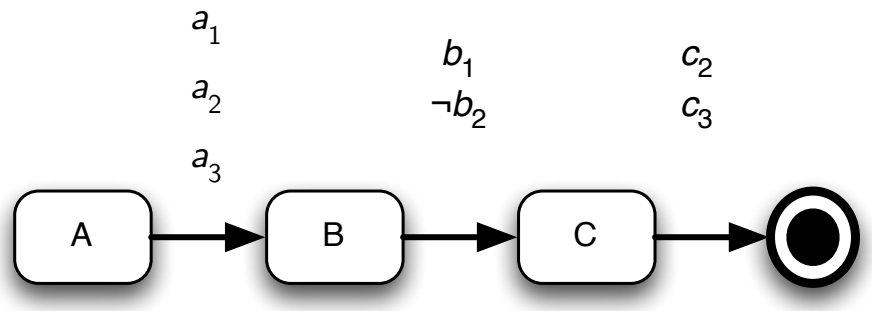
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[C,\ r_5,\ Oc_3]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled

Compensated

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

B, r_2, $Ob_2 \otimes Oc_2$ in Violated and c_2, add to Compensated

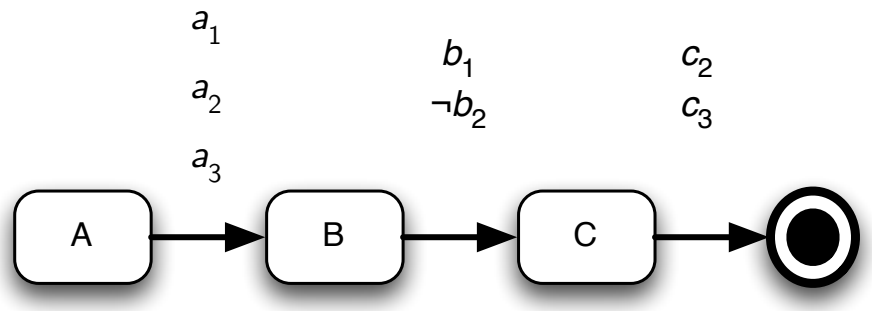
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[C,\ r_5,\ Oc_3]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled

Compensated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

Oc_3 and c_3, remove the chain from Current

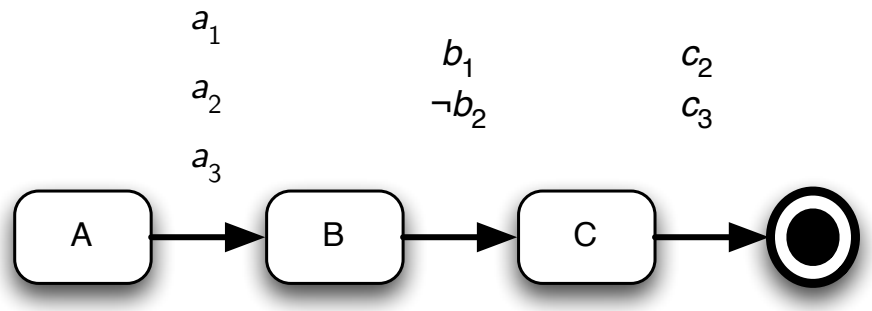
Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$
$[C,\ r_5,\ Oc_3]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled

Compensated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Checking Compliance: Example



$r_1 : a_1 \Rightarrow Ob_1 \otimes Oc_1$
$r_2 : a_2 \Rightarrow Ob_2 \otimes Oc_2$
$r_3 : a_3 \Rightarrow Ob_3 \otimes Oc_3 \otimes Od_3$
$r_4 : a_4 \Rightarrow Ob_4 \otimes Oc_4$
$r_4 : c_2 \Rightarrow Oc_3$

$Ob_3 \otimes Oc_3 \otimes Od_3$. No information about b_3 Still no information about b_3. The chain to Unfulfilled

Current
$[A,\ r_2,\ Ob_2 \otimes Oc_2]$
$[A,\ r_3,\ Ob_3 \otimes Oc_3 \otimes Od_3]$

Violated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Unfulfilled

Compensated
$[B,\ r_2,\ Ob_2 \otimes Oc_2]$

Ontology of Norms

- Persistent vs immediate obligations
  - An *immediate* obligation must be satisfied as soon as it occurs. 'When banks proceed with any wire transfer, they must transmit a message, via SWIFT, to the receiving bank requesting that the payment is made according to the instructions given.'
  - A *persistent* obligation is activated and it remain in force in the future after it has been activated.
    'A service provider must not disclose personal information without the written consent of the customer'

Immediate obligations can be used to check the 'structural compliance of a BP'

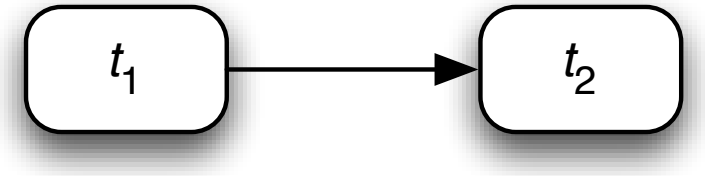
Achievement vs maintenance

- For an *achievement obligation*, a certain condition must occur at least once before the deadline
  'Customers must pay before the delivery of the good, after receiving the invoice'
- For *maintenance obligations*, a certain condition must obtain during all instants before the deadline:
  After opening a bank account, customers must keep a positive balance until bank charges are taken out.

Preemptive or Non-preemptive Obligations

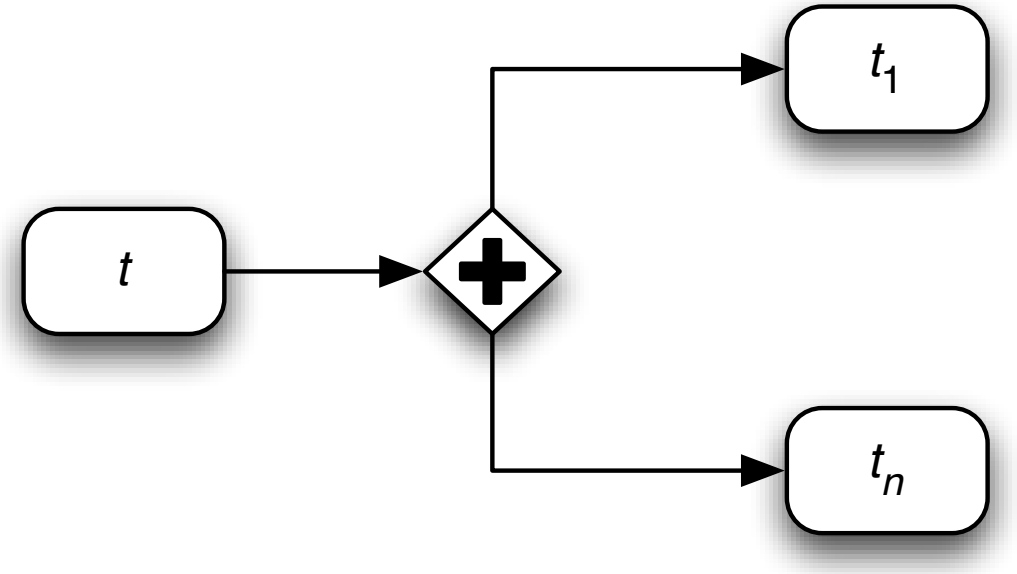
- Only for achievement obligations
- *preemptive obligations*: the fulfillment of an obligation can happen before the obligation has been triggered.
- *non preemptive obligations*: the fulfilment of an obligation can happen only after the obligation has been triggered. 'Executors and administrators of a decedent's estate will be required to give notice to each beneficiary named in the Will within 60 days after the date X of an order admitting a will to probate has been signed.'

Structural Compliance: sequence



$$t_1 \Rightarrow O^p t_2$$

Structural Compliance: AND-Split

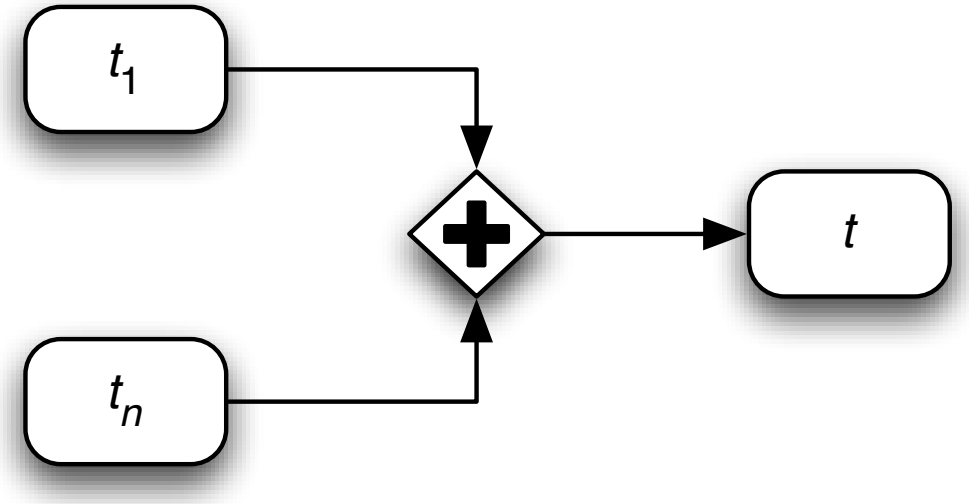


$$t \Rightarrow O^p t_1$$

$$\vdots$$

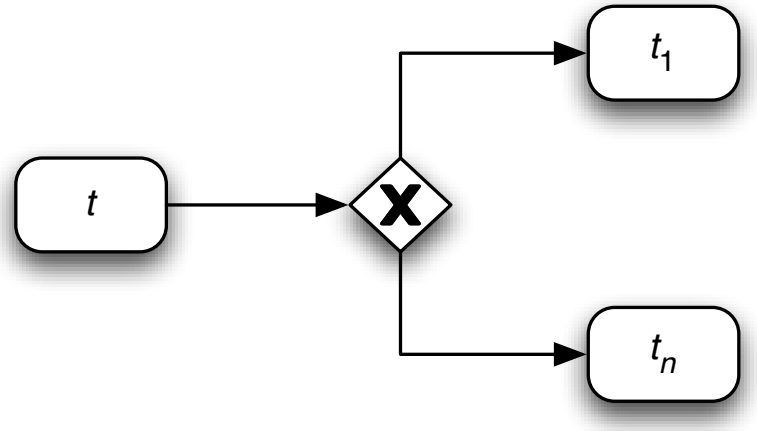
$$t \Rightarrow O^p t_n$$

Structural Compliance: AND-Join



$$t_1, \ldots, t_n \Rightarrow O^p t$$

Structural Compliance: OR-Split

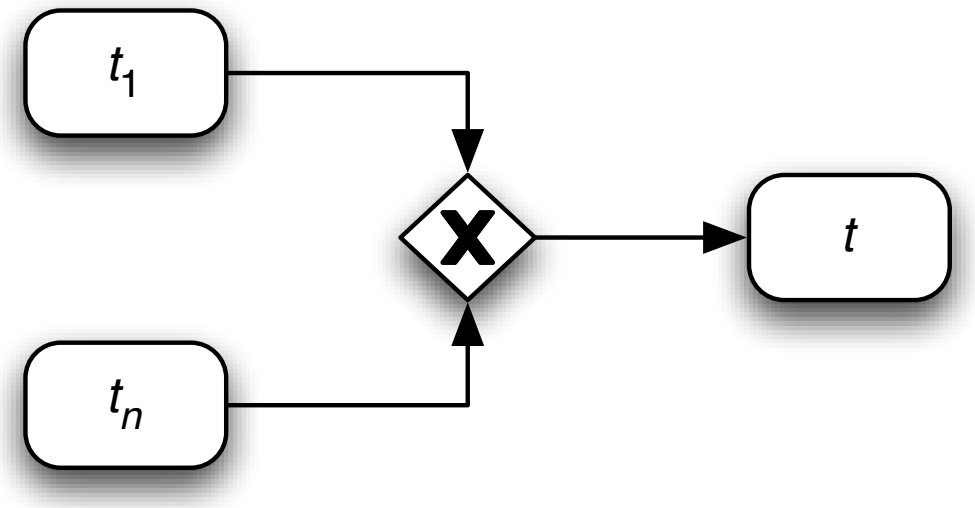


$$t \Rightarrow O^p(ORsplitID)$$
$$t_1 \Rightarrow ORsplitID$$
$$\vdots$$
$$t_n \Rightarrow ORsplitID$$

for an XOR, in addition we need

$$t_i \Rightarrow \neg t_j \quad i \neq j, 1 \leq i, j \leq n$$

Structural Compliance: OR-Joins



$$t_1 \Rightarrow O^p\, t$$

$$\vdots$$

$$t_n \Rightarrow O^p\, t$$

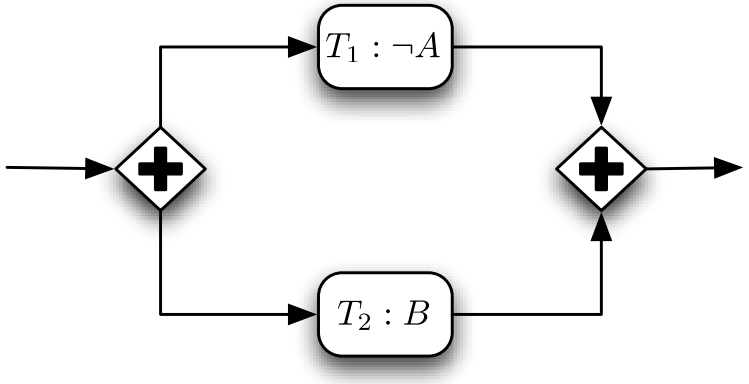
C^3: Checking Complex Compliance

NICTA

- Modifications of the basic algorithm
- C^3 seems to be computationally hard (the number of execution paths is at least exponential); is it possible to identify computationally feasible cases
  - punctual obligations: easy
  - achievement obligations without-sanction: easy
  - preemptive achievement obligations: easy
  - non-preemptive obligations: hard (interleaving of parallel subprocess, EXPTIME-hard)
  - maintenance obligations: hard ('synchronisation of sequential tasks' (EXPTIME-hard) + interleaving and synchronisation of parallel processes).

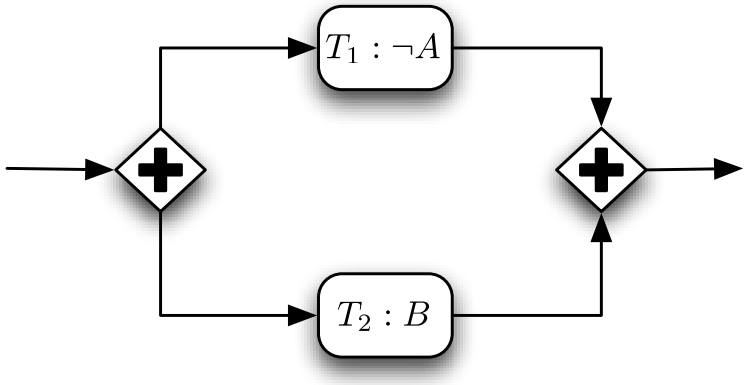
Non-preemptive Obligations

$$O^{a,np} A \otimes OB$$



Non-preemptive Obligations

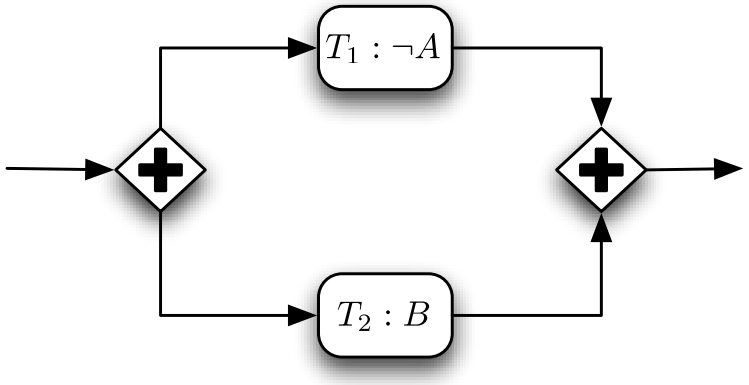
$$O^{a,np}A \otimes OB$$



$T_1; T_2$ Compliant

Non-preemptive Obligations

$$O^{a,np}A \otimes OB$$

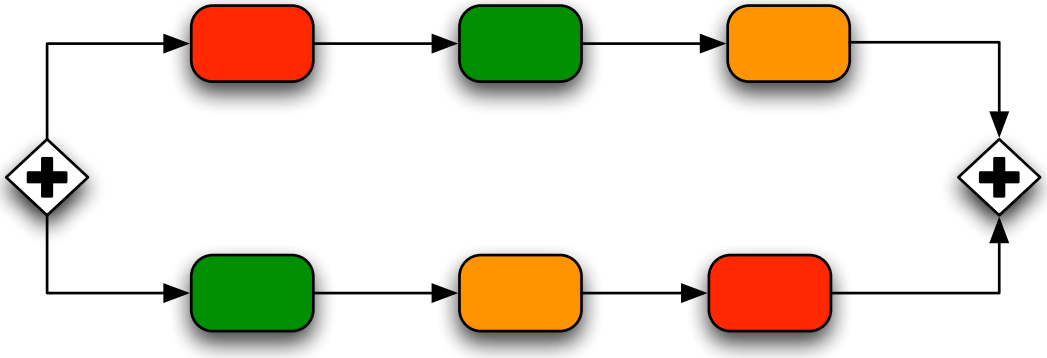


$T_1; T_2$ Compliant

$T_2; T_1$ Non-compliant

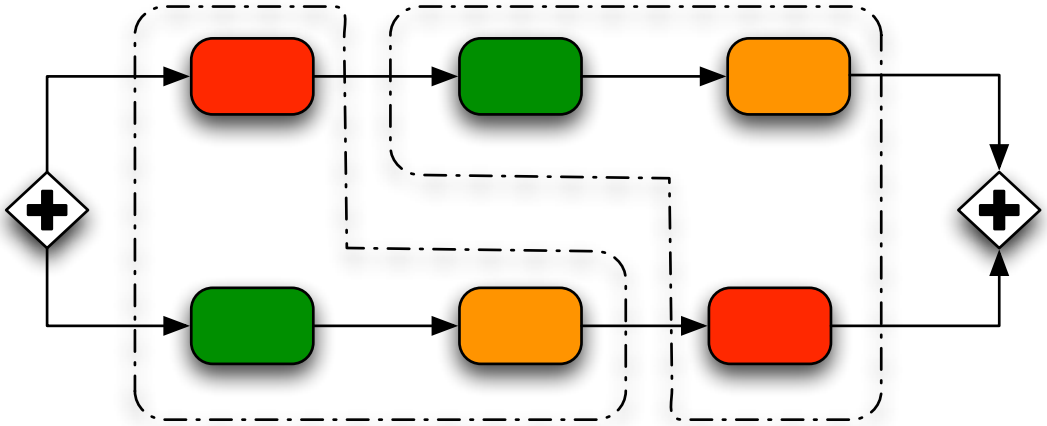
Maintenance Obligations

$O^m red$



Maintenance Obligations

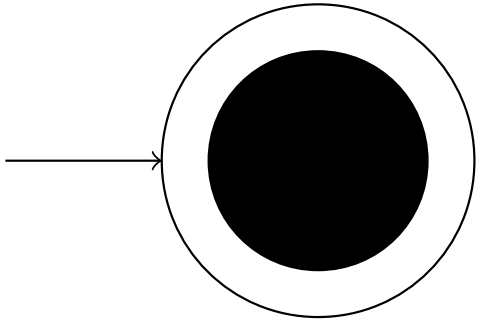
$O^m red$



Summary and Past Work

- Presented a framework to check the compliance of a business process wrt a set of norms;
- The proposed framework is computationally feasible (all the operations are in polynomial time);
- checking compliance is computationally hard.
- Extend the framework with time and deadlines and with a rich ontology of obligations: persistent vs achievement obligations.

We Are Here Now



References

Nirmit Desai, Nanjangud C. Narendra, Munindar P. Singh:.
Checking correctness of business contracts via commitments.
In *AAMAS 2008*, pp.787–794, 2008.

Andrew D. H. Farrell, Marek J. Sergot, Mathias Sallé, Claudio Bartolini.
Using the event calculus for tracking the normative state of contracts.
International Journal of Cooperative Information Systems, 14 (2–3): 99–129, 2005.

Aditya Ghose, George Koliadis.
Auditing Business Process Compliance.
In *ICSOC 2007*, LNCS 4749, pp. 169-180, Springer, 2007.

Stijn Goedertier and Jan Vanthienen.
Designing compliant business processes with obligations and permissions.
In *BPM Workshops 2006*, LNCS 4103, pp. 5–14, Springer, 2006.

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

Guido Governatori, Jörg Hoffmann, Shazia Sadiq, and Ingo Weber.
Detecting regulatory compliance for business process models through semantic annotations.
In *BPM 2008 Workshops*, LNBIP 17, pp. 5–17, Springer, 2009.

References

Guido Governatori and Zoran Milosevic.
Dealing with contract violations: formalism and domain specific language.
In *EDOC 2005*, pp. 46–57. IEEE, 2005.

Guido Governatori, Zoran Milosevic, and Shazia Sadiq.
Compliance checking between business processes and business contracts.
In *EDOC 2006*, pp. 221–232. IEEE, 2006.

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.

Guido Governatori and Antonino Rotolo.
An algorithm for business process compliance.
In Enrico Francesconi, Giovani Sartor, and Daniela Tiscornia, editors, *Jurix 2008*, pp. 186–191. IOS Press, 2008.

Guido Governatori and Antonino Rotolo.
A conceptually rich model of business process compliance.
In *APCCM 2010*, CRPIT, 2010.

Guido Governatori and Antonino Rotolo.
Norm Compliance in Business Process Modeling.
In *RuleML 2010*, LNCS 6403, Springer, 2010.

References

Guido Governatori and Shazia Sadiq.
The journey to business process compliance.
In Jorge Cardoso and Wil van der Aalst, editors, *Handbook of Research on BPM*, chapter 20, page 429–457, IGI Global, 2009.

Ying Liu, Samuel Müller, Ke Xu.
A static compliance-checking framework for business process models.
IBM Systems Journal 46: 335–362, 2007.

Ruopeng Lu, Shazia Sadiq, and Guido Governatori.
Compliance aware business process design.
In *BPM'07 Workshops*, LNCS 4928, pp. 120–131, Springer, 2007.

Ruopeng Lu, Shazia Sadiq, and Guido Governatori.
Measurement of compliance distance in business processes.
Information Systems Management, 25(4):344–355, 2008.

Stefanie Rinderle-Ma, Linh Thao Ly, Peter Dadam.
Business Process Compliance.
EMISA Forum 28(2): 24-29, 2008.

References

Anne Rozinat, Wil M. P. van der Aalst.
Conformance checking of processes based on monitoring real behavior.
Information Systems, 33: 64–95, 2008.

Shazia Sadiq and Guido Governatori.
A methodological framework for aligning business processes and regulatory compliance.
In Jan van Brocke and Michael Rosemann, editors, *Handbook of Business Process Management*, Springer, 2009.

Shazia Sadiq, Guido Governatori, and Kioumars Naimiri.
Modelling of control objectives for business process compliance.
In *BPM 2007*, LNCS 4714, 2007.

Ingo Weber, Guido Governatori, and Jörg Hoffmann.
Approximate compliance checking for annotated process models.
Information Systems Frontieres, 2009.