Specification and Analysis of Contracts
Lecture 7
Specification of ’Deontic’ Contracts Using CL

Gerardo Schneider
gerardo@ifi.uio.no
http://folk.uio.no/gerardo/

Department of Informatics,
University of Oslo

SEFM School, Oct. 27 - Nov. 7, 2008
Cape Town, South Africa
Plan of the Course

1. Introduction
2. Components, Services and Contracts
3. Background: Modal Logics 1
4. Background: Modal Logics 2
5. Deontic Logic
6. Challenges in Defining a Good Contract language
7. Specification of 'Deontic' Contracts ($\mathcal{CL}$)
8. Verification of 'Deontic' Contracts
9. Exercises
10. Exercises and Summary
Plan

1. The Contract Language $\mathcal{CL}$

2. Properties of the Language
Plan

1. The Contract Language $\mathcal{CL}$

2. Properties of the Language
Aim and Motivation

- Use deontic e-contracts to ‘rule’ services exchange (e.g., web services and component-based development)

1. Give a formal language for specifying/writing contracts
2. Analyze contracts “internally”
   - Detect contradictions/inconsistencies statically
   - Determine the obligations (permissions, prohibitions) of a signatory
   - Detect superfluous contract clauses
3. Tackle the negotiation process (automatically?)
4. Develop a theory of contracts
   - Contract composition
   - Subcontracting
   - Conformance between a contract and the governing policies
   - Meta-contracts (policies)
5. Monitor contracts
   - Run-time system to ensure the contract is respected
   - In case of contract violations, act accordingly
Aim and Motivation

- Use **deontic e-contracts** to ‘rule’ services exchange (e.g., web services and component-based development)

1. **Give a formal language** for specifying/writing contracts
2. Analyze contracts “internally”
   - Detect contradictions/inconsistencies statically
   - Determine the obligations (permissions, prohibitions) of a signatory
   - Detect superfluous contract clauses
3. **Tackle the negotiation process** (automatically?)
4. **Develop a theory of contracts**
   - Contract composition
   - Subcontracting
   - Conformance between a contract and the governing policies
   - *Meta-contracts* (policies)
5. **Monitor contracts**
   - Run-time system to ensure the contract is respected
   - In case of contract violations, act accordingly
Aim and Motivation

- Use deontic e-contracts to ‘rule’ services exchange (e.g., web services and component-based development)

1. Give a formal language for specifying/writing contracts
2. Analyze contracts “internally”
   - Detect contradictions/inconsistencies statically
   - Determine the obligations (permissions, prohibitions) of a signatory
   - Detect superfluous contract clauses

3. Tackle the negotiation process (automatically?)
4. Develop a theory of contracts
   - Contract composition
   - Subcontracting
   - Conformance between a contract and the governing policies
   - Meta-contracts (policies)

5. Monitor contracts
   - Run-time system to ensure the contract is respected
   - In case of contract violations, act accordingly
Aim and Motivation

- Use **deontic e-contracts** to ‘rule’ services exchange (e.g., web services and component-based development)

1. Give a **formal language** for specifying/writing contracts
2. **Analyze** contracts “internally”
   - Detect contradictions/inconsistencies statically
   - Determine the obligations (permissions, prohibitions) of a signatory
   - Detect superfluous contract clauses
3. **Tackle the negotiation process** (automatically?)
4. **Develop a theory of contracts**
   - Contract composition
   - Subcontracting
   - Conformance between a contract and the governing policies
   - *Meta-contracts* (policies)
5. **Monitor contracts**
   - Run-time system to ensure the contract is respected
   - In case of contract violations, act accordingly
Aim and Motivation

- Use **deontic e-contracts** to ‘rule’ services exchange (e.g., web services and component-based development)

1. **Give a formal language** for specifying/writing contracts
2. **Analyze** contracts “internally”
   - Detect contradictions/inconsistencies statically
   - Determine the obligations (permissions, prohibitions) of a signatory
   - Detect superfluous contract clauses
3. **Tackle the negotiation process** (automatically?)
4. **Develop a theory of contracts**
   - Contract composition
   - Subcontracting
   - Conformance between a contract and the governing policies
   - *Meta-contracts* (policies)
5. **Monitor contracts**
   - Run-time system to ensure the contract is respected
   - In case of contract violations, act accordingly
Aim and Motivation

- Use **deontic e-contracts** to ‘rule’ services exchange (e.g., web services and component-based development)

1. **Give a formal language** for specifying/writing contracts
2. **Analyze** contracts “internally”
   - Detect contradictions/inconsistencies statically
   - Determine the obligations (permissions, prohibitions) of a signatory
   - Detect superfluous contract clauses
3. **Tackle the negotiation process** (automatically?)
4. **Develop a theory of contracts**
   - Contract composition
   - Subcontracting
   - Conformance between a contract and the governing policies
   - *Meta-contracts* (policies)
5. **Monitor** contracts
   - Run-time system to ensure the contract is respected
   - In case of contract violations, act accordingly
A Formal Language for Contracts

- A precise and concise syntax and a formal semantics
- Expressive enough as to capture natural contract clauses
- Restrictive enough to avoid (deontic) paradoxes and be amenable to formal analysis
  - Model checking
  - Deductive verification
- Allow representation of complex clauses: conditional obligations, permissions, and prohibitions
- Allow specification of (nested) contrary-to-duty (CTD) and contrary-to-prohibition (CTP)
- CTD: when an obligation is not fulfilled
- CTP: when a prohibition is violated
- We want to combine
  - The logical approach (e.g., dynamic, temporal, deontic logic)
  - The automata-like approach (labelled Kripke structures)
A Formal Language for Contracts

- A precise and concise syntax and a formal semantics
- Expressive enough as to capture natural contract clauses
- Restrictive enough to avoid (deontic) paradoxes and be amenable to formal analysis
  - Model checking
  - Deductive verification
- Allow representation of complex clauses: conditional obligations, permissions, and prohibitions
- Allow specification of (nested) contrary-to-duty (CTD) and contrary-to-prohibition (CTP)
  - CTD: when an obligation is not fulfilled
  - CTP: when a prohibition is violated
- We want to combine
  - The logical approach (e.g., dynamic, temporal, deontic logic)
  - The automata-like approach (labelled Kripke structures)
The Contract Specification Language $\mathcal{CL}$

**Definition (\(\mathcal{CL}\))**

\[
\text{Contract} := D ; C \\
C := CO | CP | CF | C \land C | [\alpha]C | \langle\alpha\rangle C | C U C | \bigcirc C | \square C \\
CO := O(\alpha) | CO \oplus CO \\
CP := P(\alpha) | CP \oplus CP \\
CF := F(\alpha) | CF \lor [\alpha]CF
\]

- $O(\alpha)$, $P(\alpha)$, $F(\alpha)$ specify obligation, permission (rights), and prohibition (forbidden) over actions.
- $\alpha$ are actions given in the definition part $D$.
  - $+$ choice
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrency
  - $\phi$ test
- $\land$, $\lor$, and $\oplus$ are conjunction, disjunction, and exclusive disjunction.
- $[\alpha]$ and $\langle\alpha\rangle$ are the action parameterized modalities of dynamic logic.
- $U$, $\bigcirc$, and $\square$ correspond to temporal logic operators.
The Contract Specification Language $\mathcal{C}L$

**Definition ($\mathcal{C}L$)**

\[
\text{Contract} := D ; C
\]

\[
C := C_O \mid C_P \mid C_F \mid C \land C \mid [\alpha]C \mid \langle\alpha\rangle C \mid C \cup C \mid \bigcirc C \mid \Box C
\]

\[
C_O := O(\alpha) \mid C_O \oplus C_O
\]

\[
C_P := P(\alpha) \mid C_P \oplus C_P
\]

\[
C_F := F(\alpha) \mid C_F \lor [\alpha]C_F
\]

- $O(\alpha), P(\alpha), F(\alpha)$ specify obligation, permission (rights), and prohibition (forbidden) over actions
- $\alpha$ are actions given in the definition part $D$
  - $+$ choice
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrency
  - $\phi$? test
- $\land, \lor, \text{and} \oplus$ are conjunction, disjunction, and exclusive disjunction
- $[\alpha]$ and $\langle\alpha\rangle$ are the action parameterized modalities of dynamic logic
- $U, \bigcirc, \text{and} \Box$ correspond to temporal logic operators
The Contract Specification Language $\mathcal{CL}$

**Definition ($\mathcal{CL}$)**

\[
\begin{align*}
\text{Contract} & : = \ D \ ; \ C \\
C & : = \ C_{O} \ | \ C_{P} \ | \ C_{F} \ | \ C \land C \ | \ [\alpha]C \ | \ \langle \alpha \rangle C \ | \ C \cup C \ | \ \bigcirc C \ | \ \Box C \\
C_{O} & : = \ O(\alpha) \ | \ C_{O} \oplus C_{O} \\
C_{P} & : = \ P(\alpha) \ | \ C_{P} \oplus C_{P} \\
C_{F} & : = \ F(\alpha) \ | \ C_{F} \lor [\alpha]C_{F}
\end{align*}
\]

- $O(\alpha)$, $P(\alpha)$, $F(\alpha)$ specify obligation, permission (rights), and prohibition (forbidden) over actions
- $\alpha$ are actions given in the definition part $D$
  - $+$ choice
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrency
  - $\phi?$ test
- $\land$, $\lor$, and $\oplus$ are conjunction, disjunction, and exclusive disjunction
- $[\alpha]$ and $\langle \alpha \rangle$ are the action parameterized modalities of dynamic logic
- $\bigcirc, \Box, \bigcirc$ correspond to temporal logic operators
Definition ($\mathcal{CL}$)

\[\begin{align*}
\text{Contract} & := \mathcal{D} ; \mathcal{C} \\
\mathcal{C} & := \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \land \mathcal{C} \mid [\alpha] \mathcal{C} \mid \langle \alpha \rangle \mathcal{C} \mid \mathcal{C} \cup \mathcal{C} \mid \bigcirc \mathcal{C} \mid \square \mathcal{C} \\
\mathcal{C}_O & := O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P & := P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F & := F(\alpha) \mid \mathcal{C}_F \lor [\alpha] \mathcal{C}_F
\end{align*}\]

- \(O(\alpha), P(\alpha), F(\alpha)\) specify obligation, permission (rights), and prohibition (forbidden) over actions.
- \(\alpha\) are actions given in the definition part \(\mathcal{D}\).
  - + choice
  - \(\cdot\) concatenation (sequencing)
  - \& concurrency
  - \(\phi?\) test
- \(\land, \lor, \text{and} \oplus\) are conjunction, disjunction, and exclusive disjunction.
- \([\alpha]\) and \(\langle \alpha \rangle\) are the action parameterized modalities of dynamic logic.
- \(\mathcal{U}, \bigcirc, \text{and} \square\) correspond to temporal logic operators.
### Definition (CL)

**Contract** := $\mathcal{D} \ ; \ C$

$C$ := $C_O \mid C_P \mid C_F \mid C \land C \mid [\alpha]C \mid \langle \alpha \rangle C \mid C U C \mid \Diamond C \mid \Box C$

$C_O$ := $O(\alpha) \mid C_O \oplus C_O$

$C_P$ := $P(\alpha) \mid C_P \oplus C_P$

$C_F$ := $F(\alpha) \mid C_F \lor [\alpha]C_F$

- $O(\alpha)$, $P(\alpha)$, $F(\alpha)$ specify obligation, permission (rights), and prohibition (forbidden) over actions
- $\alpha$ are actions given in the definition part $\mathcal{D}$
  - $+$ choice
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrency
  - $\phi?$ test
- $\land$, $\lor$, and $\oplus$ are conjunction, disjunction, and exclusive disjunction
- $[\alpha]$ and $\langle \alpha \rangle$ are the action parameterized modalities of dynamic logic
- $U$, $\Diamond$, and $\Box$ correspond to temporal logic operators
Tests as actions: \( \phi \)?

- The behaviour of a test is like a *guard*; e.g. \( \phi? \cdot a \) if the test succeeds then action \( a \) is performed
- Tests are used to model implication: \([\phi?]C\) is the same as \( \varphi \Rightarrow C \)

Action negation \( \overline{\alpha} \)

- It represents all immediate traces that take us outside the trace of \( \alpha \)
- Involves the use of a *canonic form* of actions
- E.g.: consider two atomic actions \( a \) and \( b \) then \( \overline{a \cdot b} \) is \( b + a \cdot a \)
Test and Negation

- **Tests** as actions: $\phi$?
  - The behaviour of a test is like a *guard*; e.g. $\phi \cdot a$ if the test succeeds then action $a$ is performed
  - Tests are used to model implication: $[\phi?]C$ is the same as $\phi \Rightarrow C$

- **Action negation** $\overline{\alpha}$
  - It represents all immediate traces that take us outside the trace of $\alpha$
  - Involves the use of a *canonic form* of actions
  - E.g.: consider two atomic actions $a$ and $b$ then $a \cdot b$ is $b + a \cdot a$
Actions

Concurrent actions

- \(a \& b\)
- “The client must pay immediately, or the client must notify the service provider by sending an e-mail specifying that he delays the payment”

\[ O(p) \oplus O(d \& n) \]

- \(O(d \& n) \equiv O(d) \land O(n)\)
- Action algebra enriched with a conflict relation to represent incompatible actions
  - \(a = \) “increase Internet traffic” and \(b = \) “decrease Internet traffic”
  - \(a \#_c b\)
  - \(O(a) \land O(b)\) gives an inconsistency
Actions

Concurrent actions

- \( a \& b \)
- “The client must pay immediately, or the client must notify the service provider by sending an e-mail specifying that he delays the payment”

\[
O(p) \oplus O(d \& n)
\]

- \( O(d \& n) \equiv O(d) \land O(n) \)
- Action algebra enriched with a conflict relation to represent incompatible actions
  - \( a = \text{“increase Internet traffic”} \) and \( b = \text{“decrease Internet traffic”} \)
  - \( a \not\equiv b \)
  - \( O(a) \land O(b) \) gives an inconsistency
More on the Contract Language
CTD and CTP

- Expressing contrary-to-duty (CTD)

\[ O_C(\alpha) = O(\alpha) \land [\overline{\alpha}]C \]

Example: "[...] the client must immediately lower the Internet traffic to the low level, and pay. If the client does not lower the Internet traffic immediately, then the client will have to pay three times the price"

In CL: \[ \Box (O_C(l) \land [l]\Box (O(p \& p \& p))) \]

where \( C = \Box O(p \& p \& p) \)
More on the Contract Language

CTD and CTP

- Expressing *contrary-to-duty* (CTD)
  \[ O_C(\alpha) = O(\alpha) \land [\overline{\alpha}]C \]

- Expressing *contrary-to-prohibition* (CTP)
  \[ F_C(\alpha) = F(\alpha) \land [\alpha]C \]
More on the Contract Language

CTD and CTP

- Expressing **contrary-to-duty** (CTD)

\[ O_\alpha(\alpha) = O(\alpha) \land [\overline{\alpha}]C \]

- Expressing **contrary-to-prohibition** (CTP)

\[ F_\alpha(\alpha) = F(\alpha) \land [\alpha]C \]

**Example**

“[...] the client must immediately lower the Internet traffic to the low level, and pay. If the client does not lower the Internet traffic immediately, then the client will have to pay three times the price”

In CL:

\[ \Box (O_\alpha(I) \land [I]\Diamond (O(p&p))) \]

where \[ C = \Diamond O(p&p&p) \]
A first semantics given through a translation into a variant of μ-calculus ($C\mu$)
- A Kripke-like modal semantics have been developed recently

Why μ-calculus?
- μ-calculus is a combination of propositional logic, the action parameterized modal operator $[a]$, and the fix point constructions
- Expressive – embeds most of the used temporal and process logics
- Well studied – has a complete axiomatic system and a complete proof system
- Very efficient algorithms for model checking
- Mathematically well founded in the results on fix points (Tarski, Knaster, Kleene, et al.)
- The modal variant of μ-calculus is based on actions (labels)
Definition

The syntax of the $\mathcal{C}_\mu$ calculus is defined as follows:

$$\phi := P \mid Z \mid P_c \mid \top \mid \neg \phi \mid \phi \land \phi \mid [\gamma] \phi \mid \mu Z. \phi(Z)$$

Main differences with respect to the classical $\mu$-calculus:

1. $P_c$ is set of propositional constants $O_a$ and $F_a$, one for each basic action $a$
   - Semantic restriction: $\|F_a\|_\gamma \cap \|O_a\|_\gamma = \emptyset$, $\forall a \in L$
2. Multisets of basic actions: i.e. $\gamma = \{a, a, b\}$ is a label
**Definition**

The syntax of the $C\mu$ calculus is defined as follows:

$$\phi ::= P \mid Z \mid P_c \mid \top \mid \neg\phi \mid \phi \land \phi \mid [\gamma]\phi \mid \mu Z.\phi(Z)$$

Main differences with respect to the classical $\mu$-calculus:

1. $P_c$ is set of propositional constants $O_a$ and $F_a$, one for each basic action $a$
   - Semantic restriction: $\|F_a\|_V \cap \|O_a\|_V = \emptyset$, $\forall a \in \mathcal{L}$

2. Multisets of basic actions: i.e. $\gamma = \{a, a, b\}$ is a label
Obligation

\[ f^T(O(a \& b)) = \langle \{a, b\}\rangle (O_a \land O_b) \]
Obligation

\[ f^I (O(a\&b)) = \langle \{a, b\} \rangle (O_a \land O_b) \]
We would like to have a compositional semantics and preserve the intuitive properties of obligations, permissions and prohibitions.

Also: get rid of paradoxes!

Conjunction in dynamic logic is a branching.

\( \parallel O(a) \land O(b) \parallel \) should be defined as \( \parallel O(a) \parallel \) and \( \parallel O(b) \parallel \). How to enforce it?

\( \parallel P(\alpha \beta) \parallel \equiv \parallel P(\alpha) \land \langle \alpha \rangle P(\beta) \parallel \)

\( O(a \& b) \equiv O(a) \land O(b) \). Solution:

We will add some equivalences and rewriting rules to enforce the above.
CL Semantics
Difficulties in the Encoding

- We would like to have a compositional semantics and preserve the intuitive properties of obligations, permissions and prohibitions
- Also: get rid of paradoxes!

Not easy!
We would like to have a compositional semantics and preserve the intuitive properties of obligations, permissions and prohibitions.

Also: get rid of paradoxes!

Not easy!

Conjunction in dynamic logic is a branching.

What is the semantics of $O(a) \land O(b)$?

$\|O(a) \land O(b)\|$ should be defined as $\|O(a)\|$ and $\|O(b)\|$.

How to enforce it?

How to enforce some properties?

$\|P(\alpha \beta)\| \equiv \|P(\alpha) \land \langle \alpha \rangle P(\beta)\|$

$O(a \& b) \equiv O(a) \land O(b)$
We would like to have a compositional semantics and preserve the intuitive properties of obligations, permissions and prohibitions.

Also: get rid of paradoxes!

Not easy!

Conjunction in dynamic logic is a branching.

What is the semantics of $O(a) \land O(b)$?

- $\llbracket O(a) \land O(b) \rrbracket$ should be defined as $\llbracket O(a) \rrbracket$ and $\llbracket O(b) \rrbracket$
- How to enforce it?

How to enforce some properties?

- $\llbracket P(\alpha \beta) \rrbracket \equiv \llbracket P(\alpha) \land \langle \alpha \rangle P(\beta) \rrbracket$
- $O(a \& b) \equiv O(a) \land O(b)$

Solution

We will add some equivalences and rewriting rules to enforce the above.
### Compositional Rules

1. \( O(\alpha + \beta) \equiv O(\alpha) \oplus O(\beta) \)
2. \( O(a \& b) \equiv O(a) \land O(b) \)
3. \( O(\alpha \beta) \equiv O(\alpha) \land [\alpha]O(\beta) \)
4. \( P(\alpha + \beta) \equiv P(\alpha) \oplus P(\beta) \)
5. \( P(\alpha \beta) \equiv P(\alpha) \land \langle \alpha \rangle P(\beta) \)
6. \( F(\alpha \beta) \equiv F(\alpha) \lor [\alpha]F(\beta) \)

- Some of the above are intended to force “common sense” relationship
  - If we were to define an axiomatic system, we would aim the above to be axioms or theorems
- Concurrent actions are compositional only under obligation —No similar rules for \( F \) and \( P \)
### Rewriting Rules for Obligation

<table>
<thead>
<tr>
<th>Rule</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>( O(a) \land O(b) \leadsto O(a &amp; b) )</td>
</tr>
<tr>
<td>(2)</td>
<td>( O(a) \land O(a &amp; b) \leadsto O(a &amp; b) )</td>
</tr>
<tr>
<td>(3)</td>
<td>( O(a) \land (O(a) \oplus O(b)) \leadsto O(a) )</td>
</tr>
<tr>
<td>(4)</td>
<td>( O(a) \land O(a) \leadsto O(a) )</td>
</tr>
<tr>
<td>(5)</td>
<td>( O(a) \oplus O(a) \leadsto O(a) )</td>
</tr>
<tr>
<td>(6)</td>
<td>( O(c) \land (O(a) \oplus O(b)) \leadsto (O(c) \land O(a)) \oplus (O(c) \land O(b)) )</td>
</tr>
<tr>
<td>(7)</td>
<td>( (\oplus_i O(a_i)) \land (\oplus_j O(b_j)) \leadsto \oplus_{i,j} (O(a_i) \land O(b_j)) \quad a_i \neq b_j )</td>
</tr>
</tbody>
</table>

- Rules (1)-(3): guided by intuition
- Rules (4)-(5): usual contraction rules
- Rules (6)-(7): distributivity of conjunction over the exclusive disjunction
Definition (The Semantic Encoding)

1. \( f^T(O(\&_{i=1}^n a_i)) = \langle \{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^n O_{a_i}) \)
2. \( f^T(C_O \oplus C_O) = f^T(C_O) \wedge f^T(C_O) \)
3. \( f^T(P(\&_{i=1}^n a_i)) = \langle \{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^n \neg F_{a_i}) \)
4. \( f^T(C_P \oplus C_P) = f^T(C_P) \wedge f^T(C_P) \)
5. \( f^T(F(\&_{i=1}^n a_i)) = [\{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^n F_{a_i}) \)
6. \( f^T(F(\delta) \lor [\beta]F(\delta)) = f^T(F(\delta)) \lor f^T([\beta]F(\delta)) \)
7. \( f^T(C_1 \wedge C_2) = f^T(C_1) \wedge f^T(C_2) \)
8. \( f^T(\circ C) = [\text{any}]f^T(C) \)
9. \( f^T(C_1 \cup C_2) = \mu Z.f^T(C_2) \lor (f^T(C_1) \wedge [\text{any}]Z \wedge \langle \text{any} \rangle \top) \)
10. \( f^T([\&_{i=1}^n a_i]C) = [\{a_1, \ldots, a_n\}]f^T(C) \)
11. \( f^T([[(\&_{i=1}^n a_i)\alpha]C) = [\{a_1, \ldots, a_n\}]f^T([\alpha]C) \)
12. \( f^T([\alpha + \beta]C) = f^T([\alpha]C) \wedge f^T([\beta]C) \)
13. \( f^T([\varphi?]C) = f^T(\varphi) \implies f^T(C) \)
Example

- \( f^T(\bigwedge_{i=1}^{n} a_i) = \langle\{a_1, \ldots, a_n\}\rangle(\bigwedge_{i=1}^{n} O a_i) \)

  - “The Provider is obliged to provide internet and telephony services (at the same time)”: 

    \[ f^T(O(a\&b)) = \langle\{a, b\}\rangle(O a \land O b) \]

- \( f^T(F(\bigwedge_{i=1}^{n} a_i)) = [\{a_1, \ldots, a_n\}] (\bigwedge_{i=1}^{n} F a_i) \)

  - “It is forbidden to send private information”

    \[ f^T(F(a)) = [a]F a \]

- \( f^T(P(\bigwedge_{i=1}^{n} a_i)) = \langle\{a_1, \ldots, a_n\}\rangle(\bigwedge_{i=1}^{n} \neg F a_i) \)

  - “It is permitted to receive an acknowledgement”

    \[ f^T(P(a)) = \langle a\rangle \neg F a \]
**Example**

- \( f^T(O(\&_{i=1}^n a_i)) = \langle \{a_1, \ldots, a_n\}\rangle(\&_{i=1}^n O_{a_i}) \)
  - “The Provider is obliged to provide internet and telephony services (at the same time)”: 
    \[ f^T(O(a\&b)) = \langle \{a, b\}\rangle(O_a \land O_b) \]
- \( f^T(F(\&_{i=1}^n a_i)) = [\{a_1, \ldots, a_n\}\](\&_{i=1}^n F_{a_i}) \)
  - “It is forbidden to send private information”
    \[ f^T(F(a)) = [a]F_a \]
- \( f^T(P(\&_{i=1}^n a_i)) = \langle \{a_1, \ldots, a_n\}\rangle(\&_{i=1}^n F_{a_i}) \)
  - “It is permitted to receive an acknowledgement”
    \[ f^T(P(a)) = \langle a\rangle\neg F_a \]
Example

- \( f^T(O(\&_{i=1}^n a_i)) = \langle \{a_1, \ldots, a_n\} \rangle (\&_{i=1}^n O a_i) \)
  - "The Provider is obliged to provide internet and telephony services (at the same time)":
    \[
    f^T(O(a \& b)) = \langle \{a, b\} \rangle (O_a \land O_b)
    \]

- \( f^T(F(\&_{i=1}^n a_i)) = [\{a_1, \ldots, a_n\}] (\&_{i=1}^n F a_i) \)
  - "It is forbidden to send private information"
    \[
    f^T(F(a)) = [a]F_a
    \]

- \( f^T(P(\&_{i=1}^n a_i)) = \langle \{a_1, \ldots, a_n\} \rangle (\&_{i=1}^n \neg F a_i) \)
  - "It is permitted to receive an acknowledgement"
    \[
    f^T(P(a)) = \langle a \rangle \neg F_a
    \]
Example

**Contrary-to-duty (CTD):**  $O_{O(b)}(a) = O(a) \land [\overline{a}]O(b)$

Applying the semantic encoding:

$$f^T(O_{O(b)}(a)) = \langle a \rangle O_a \land [\overline{a}] \langle b \rangle O_b$$
Example

- **Contrary-to-duty (CTD):** \( O_{O(b)}(a) = O(a) \land [\bar{a}]O(b) \)

Applying the semantic encoding:

\[
f^T (O_{O(b)}(a)) = \langle a \rangle O_a \land [\bar{a}]\langle b \rangle O_b
\]
Contrary-to-duty (CTD): $O_{O(b)}(a) = O(a) \land [\overline{a}]O(b)$

Applying the semantic encoding:

$$f^T(O_{O(b)}(a)) = \langle a \rangle O_a \land [\overline{a}]\langle b \rangle O_b$$

Contrary-to-prohibition (CTP): $F_{O(b)}(a) = F(a) \land [a]O(b)$

Applying the semantic encoding:

$$f^T(F_{O(b)}(a)) = [a]F_a \land [a]\langle b \rangle O_b$$
Plan

1. The Contract Language $\mathcal{CL}$

2. Properties of the Language
The following paradoxes are avoided in $CL$:

- Ross’s paradox
- The Free Choice Permission paradox
- Sartre’s dilemma
- The Good Samaritan paradox
- Chisholm’s paradox
- The Gentle Murderer paradox
Ross's paradox

1. It is obligatory that one mails the letter
2. It is obligatory that one mails the letter or one destroys the letter

In Standard Deontic Logic (SDL) these are expressed as:

1. $O(p)$
2. $O(p \lor q)$

Problem

In SDL one can infer that $O(p) \implies O(p \lor q)$
Ross’s paradox

1. It is obligatory that one mails the letter
2. It is obligatory that one mails the letter or one destroys the letter

In Standard Deontic Logic (SDL) these are expressed as:

1. $O(p)$
2. $O(p \lor q)$

Problem

In SDL one can infer that $O(p) \Rightarrow O(p \lor q)$
Ross’s paradox

1. It is obligatory that one mails the letter
2. It is obligatory that one mails the letter or one destroys the letter

In Standard Deontic Logic (SDL) these are expressed as:
1. \( O(p) \)
2. \( O(p \lor q) \)

**Problem**

In SDL one can infer that \( O(p) \implies O(p \lor q) \)

**Avoided in CL**

**Proof Sketch:**
- \( f^T(O(a)) = \langle a \rangle O_a \)
- \( O(a + b) \equiv O(a) \oplus O(b) \quad f^T = \langle a \rangle O_a \land \langle b \rangle O_b \)
- \( \langle a \rangle O_a \not\Rightarrow \langle a \rangle O_a \land \langle b \rangle O_b \)
Chisholm’s Paradox

1. John ought to go to the party.
2. If John goes to the party then he ought to tell them he is coming.
3. If John does not go to the party then he ought not to tell them he is coming.
4. John does not go to the party.

In Standard Deontic Logic (SDL) these are expressed as:

1. $O(p)$
2. $O(p \Rightarrow q)$
3. $\neg p \Rightarrow O(\neg q)$
4. $\neg p$

**Problem**

The problem is that in SDL one can infer $O(q) \land O(\neg q)$ (due to 2)
Chisholm’s Paradox

1. John ought to go to the party.
2. If John goes to the party then he ought to tell them he is coming.
3. If John does not go to the party then he ought not to tell them he is coming.
4. John does not go to the party.

In Standard Deontic Logic (SDL) these are expressed as:

1. \( O(p) \)
2. \( O(p \Rightarrow q) \)
3. \( \neg p \Rightarrow O(\neg q) \)
4. \( \neg p \)

Problem

The problem is that in SDL one can infer \( O(q) \land O(\neg q) \) (due to 2)
Avoided in $\mathcal{CL}$

Expressed in $\mathcal{CL}$ as:

1. $O(a)$
2. $[a]O(b)$
3. $[\overline{a}]O(\overline{b})$

- (1) and (3) give the CTD formula $O_\varphi(a)$ of $\mathcal{CL}$ where $\varphi = O(\overline{b})$
- In $\mathcal{CL}$ $O(b)$ and $O(\overline{b})$ cannot hold in the same world
  - $O(b)$ holds only after doing action $a$, where $O(\overline{b})$ holds only after doing the contradictory action $\overline{a}$
Properties of the contract language (II)

Theorem

The following hold in CL:

- $P(\alpha) \equiv \neg F(\alpha)$
- $O(\alpha) \Rightarrow P(\alpha)$
- $P(a) \nRightarrow P(a \& b)$
- $F(a) \nRightarrow F(a \& b)$
- $F(a \& b) \nRightarrow F(a)$
- $P(a \& b) \nRightarrow P(a)$
We have seen...

- $\mathcal{CL}$: A formal language to write contracts
- The formal semantics given through an encoding into a $\mu$-calculus variant
- It avoids the most important paradoxes of deontic logic
- Does not address all the issues of the 'ideal' language presented in last lecture
Final Remarks

We have seen...

- $\mathcal{CL}$: A formal language to write contracts
- The formal semantics given through an encoding into a $\mu$-calculus variant
- It avoids the most important paradoxes of deontic logic
- Does not address all the issues of the 'ideal' language presented in last lecture

Next lecture

- We will see how to model check contracts written in $\mathcal{CL}$
Further Reading