Specification and Analysis of Contracts
Lecture 7
Specification of 'Deontic' Contracts Using $\mathcal{CL}$

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

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 ($\mathcal{CL}$)

$$
\begin{array}{rcl}
\mathcal{C}ontract & := & \mathcal{D} \; ; \; \mathcal{C} \\
\mathcal{C} & := & \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle\alpha\rangle\mathcal{C} \mid \mathcal{C}\,\mathcal{U}\,\mathcal{C} \mid \bigcirc\,\mathcal{C} \mid \Box\mathcal{C} \\
\mathcal{C}_O & := & O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P & := & P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F & := & F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F
\end{array}
$$

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

The Contract Specification Language $\mathcal{CL}$

Definition ($\mathcal{CL}$)

$$
\begin{array}{rcl}
\mathcal{C}ontract & := & \mathcal{D}\ ;\ \mathcal{C} \\
\mathcal{C} & := & \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle\alpha\rangle\mathcal{C} \mid \mathcal{C}\,\mathcal{U}\,\mathcal{C} \mid \bigcirc\,\mathcal{C} \mid \Box\mathcal{C} \\
\mathcal{C}_O & := & O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P & := & P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F & := & F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F
\end{array}
$$

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

Definition ($\mathcal{CL}$)

$$
\begin{array}{rcl}
\mathcal{C}ontract & := & \mathcal{D} \; ; \; \mathcal{C} \\
\mathcal{C} & := & \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle\alpha\rangle\mathcal{C} \mid \mathcal{C}\,\mathcal{U}\,\mathcal{C} \mid \bigcirc\,\mathcal{C} \mid \Box\mathcal{C} \\
\mathcal{C}_O & := & O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P & := & P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F & := & F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F
\end{array}
$$

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

The Contract Specification Language $\mathcal{CL}$

Definition ($\mathcal{CL}$)

$$
\begin{array}{rcl}
\mathcal{C}ontract & := & \mathcal{D} \; ; \; \mathcal{C} \\
\mathcal{C} & := & \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle\alpha\rangle\mathcal{C} \mid \mathcal{C}\,\mathcal{U}\,\mathcal{C} \mid \bigcirc\,\mathcal{C} \mid \Box\mathcal{C} \\
\mathcal{C}_O & := & O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P & := & P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F & := & F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F
\end{array}
$$

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

The Contract Specification Language $\mathcal{CL}$

Definition ($\mathcal{CL}$)

$$
\begin{array}{rcl}
\mathcal{C}ontract & := & \mathcal{D}\ ;\ \mathcal{C} \\
\mathcal{C} & := & \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle\alpha\rangle\mathcal{C} \mid \mathcal{C}\,\mathcal{U}\,\mathcal{C} \mid \bigcirc\,\mathcal{C} \mid \Box\mathcal{C} \\
\mathcal{C}_O & := & O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P & := & P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F & := & F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F
\end{array}
$$

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

- Tests as actions: $\phi?$
  - The behaviour of a test is like a *guard*; e.g. $\varphi? \cdot a$ *if the test succeeds then action a is performed*
  - Tests are used to model implication: $[\varphi?]\mathcal{C}$ is the same as $\varphi \Rightarrow \mathcal{C}$
- Action negation $\overline{\alpha}$
  - It represents all immediate traces that take us outside the trace of α
  - Involves the use of a *canonic form* of actions
  - E.g.: consider two atomic actions a and b then $\overline{a \cdot b}$ is $b + a \cdot a$

- Tests as actions: $\phi?$
  - The behaviour of a test is like a *guard*; e.g. $\varphi? \cdot a$ *if the test succeeds then action a is performed*
  - Tests are used to model implication: $[\varphi?]\mathcal{C}$ is the same as $\varphi \Rightarrow \mathcal{C}$
- Action negation $\overline{\alpha}$
  - It represents all immediate traces that take us outside the trace of α
  - Involves the use of a *canonic form* of actions
  - E.g.: consider two atomic actions a and b then $\overline{a \cdot b}$ is $b + a \cdot a$

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

$$O(p) \oplus O(d \& n)$$

- $O(d \& n) \equiv O(d) \wedge O(n)$
- Action algebra enriched with a conflict relation to represent incompatible actions
  - $a =$ "increase Internet traffic" and $b =$ "decrease Internet traffic"
    - $a \#_{\mathcal{C}} b$
    - $O(a) \wedge O(b)$ gives an inconsistency

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

$$O(p) \oplus O(d \& n)$$

- $O(d \& n) \equiv O(d) \wedge O(n)$
- Action algebra enriched with a conflict relation to represent incompatible actions
  - $a =$ "increase Internet traffic" and $b =$ "decrease Internet traffic"
    - $a \#_{\mathcal{C}} b$
    - $O(a) \wedge O(b)$ gives an inconsistency

- Expressing contrary-to-duty (CTD)

$$O_{\mathcal{C}}(\alpha) = O(\alpha) \wedge [\overline{\alpha}]\mathcal{C}$$

More on the Contract Language
CTD and CTP

- Expressing contrary-to-duty (CTD)

$$O_{\mathcal{C}}(\alpha) = O(\alpha) \wedge [\overline{\alpha}]\mathcal{C}$$

- Expressing contrary-to-prohibition (CTP)

$$F_{\mathcal{C}}(\alpha) = F(\alpha) \wedge [\alpha]\mathcal{C}$$

More on the Contract Language
CTD and CTP

- Expressing contrary-to-duty (CTD)

$$O_{\mathcal{C}}(\alpha) = O(\alpha) \wedge [\overline{\alpha}]\mathcal{C}$$

- Expressing contrary-to-prohibition (CTP)

$$F_{\mathcal{C}}(\alpha) = F(\alpha) \wedge [\alpha]\mathcal{C}$$

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 $\mathcal{CL}$: $\Box(O_{\mathcal{C}}(l) \wedge [l]\Diamond(O(p\&p)))$

where $\mathcal{C} = \Diamond O(p\&p\&p)$

$\mathcal{CL}$ Semantics

- A first semantics given through a translation into a variant of μ-calculus ($\mathcal{C}\mu$)
  - 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:

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

Main differences with respect to the classical μ-calculus:

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

Definition

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

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

Main differences with respect to the classical μ-calculus:

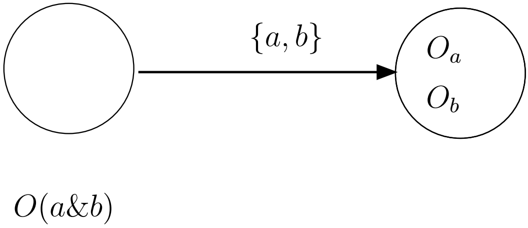
1. P_c is set of propositional constants O_a and $\mathcal{F}_a$, one for each basic action a
   - Semantic restriction: $\|\mathcal{F}_a\|^{\mathcal{T}}_{\mathcal{V}} \cap \|O_a\|^{\mathcal{T}}_{\mathcal{V}} = \emptyset, \quad \forall a \in \mathcal{L}$
2. Multisets of basic actions: i.e. $\gamma =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

- Obligation

$$f^{\mathcal{T}}(O(a\&b)) = \langle\{a, b\}\rangle(O_a \wedge O_b)$$

- Obligation

$$f^{\mathcal{T}}(O(a\&b)) = \langle\{a,b\}\rangle(O_a \wedge O_b)$$



$\mathcal{CL}$ 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!

$\mathcal{CL}$ 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 $O(a) \wedge O(b)$?
  - $\|O(a) \wedge O(b)\|$ should be defined as $\|O(a)\|$ **and** $\|O(b)\|$
  - How to enforce it?
- How to enforce some properties?
  - $\|P(\alpha\beta)\| \equiv \|P(\alpha) \wedge \langle\alpha\rangle P(\beta)\|$
  - $O(a\&b) \equiv O(a) \wedge O(b)$

- 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 $O(a) \wedge O(b)$?
  - $\|O(a) \wedge O(b)\|$ should be defined as $\|O(a)\|$ **and** $\|O(b)\|$
  - How to enforce it?
- How to enforce some properties?
  - $\|P(\alpha\beta)\| \equiv \|P(\alpha) \wedge \langle\alpha\rangle P(\beta)\|$
  - $O(a\&b) \equiv O(a) \wedge O(b)$

Solution

We will add some equivalences and rewriting rules to enforce the above

Compositional Rules

$$
\begin{array}{lrcl}
(1) & O(\alpha + \beta) & \equiv & O(\alpha) \oplus O(\beta) \\
(2) & O(a\&b) & \equiv & O(a) \wedge O(b) \\
(3) & O(\alpha\beta) & \equiv & O(\alpha) \wedge [\alpha]O(\beta) \\
(4) & P(\alpha + \beta) & \equiv & P(\alpha) \oplus P(\beta) \\
(5) & P(\alpha\beta) & \equiv & P(\alpha) \wedge \langle\alpha\rangle P(\beta) \\
(6) & F(\alpha\beta) & \equiv & F(\alpha) \vee [\alpha]F(\beta)
\end{array}
$$

- 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

$\mathcal{CL}$ Semantics

Pre-processing

Rewriting Rules for Obligation

$$
\begin{array}{lrcl}
(1) & O(a) \wedge O(b) & \rightsquigarrow & O(a\&b) \\
(2) & O(a) \wedge O(a\&b) & \rightsquigarrow & O(a\&b) \\
(3) & O(a) \wedge (O(a) \oplus O(b)) & \rightsquigarrow & O(a) \\
(4) & O(a) \wedge O(a) & \rightsquigarrow & O(a) \\
(5) & O(a) \oplus O(a) & \rightsquigarrow & O(a) \\
(6) & O(c) \wedge (O(a) \oplus O(b)) & \rightsquigarrow & (O(c) \wedge O(a)) \oplus (O(c) \wedge O(b)) \\
(7) & (\oplus_i O(a_i)) \wedge (\oplus_j O(b_j)) & \rightsquigarrow & \oplus_{i,j}(O(a_i) \wedge O(b_j)) \quad a_i \neq b_j
\end{array}
$$

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

$\mathcal{CL}$ Semantics

Definition (The Semantic Encoding)

$$
\begin{array}{lrcl}
(1) & f^{\mathcal{T}}(O(\&_{i=1}^{n} a_i)) & = & \langle\{a_1,\ldots,a_n\}\rangle(\wedge_{i=1}^{n} O_{a_i}) \\
(2) & f^{\mathcal{T}}(\mathcal{C}_O \oplus \mathcal{C}_O) & = & f^{\mathcal{T}}(\mathcal{C}_O) \wedge f^{\mathcal{T}}(\mathcal{C}_O) \\
(3) & f^{\mathcal{T}}(P(\&_{i=1}^{n} a_i)) & = & \langle\{a_1,\ldots,a_n\}\rangle(\wedge_{i=1}^{n} \neg\mathcal{F}_{a_i}) \\
(4) & f^{\mathcal{T}}(\mathcal{C}_P \oplus \mathcal{C}_P) & = & f^{\mathcal{T}}(\mathcal{C}_P) \wedge f^{\mathcal{T}}(\mathcal{C}_P) \\
(5) & f^{\mathcal{T}}(F(\&_{i=1}^{n} a_i)) & = & [\{a_1,\ldots,a_n\}](\wedge_{i=1}^{n} \mathcal{F}_{a_i}) \\
(6) & f^{\mathcal{T}}(F(\delta) \vee [\beta]F(\delta)) & = & f^{\mathcal{T}}(F(\delta)) \vee f^{\mathcal{T}}([\beta]F(\delta)) \\
(7) & f^{\mathcal{T}}(\mathcal{C}_1 \wedge \mathcal{C}_2) & = & f^{\mathcal{T}}(\mathcal{C}_1) \wedge f^{\mathcal{T}}(\mathcal{C}_2) \\
(8) & f^{\mathcal{T}}(\bigcirc\mathcal{C}) & = & [\mathbf{any}]f^{\mathcal{T}}(\mathcal{C}) \\
(9) & f^{\mathcal{T}}(\mathcal{C}_1 \,\mathcal{U}\, \mathcal{C}_2) & = & \mu Z.f^{\mathcal{T}}(\mathcal{C}_2) \vee (f^{\mathcal{T}}(\mathcal{C}_1) \wedge [\mathbf{any}]Z \wedge \langle\mathbf{any}\rangle\top) \\
(10) & f^{\mathcal{T}}([\&_{i=1}^{n} a_i]\mathcal{C}) & = & [\{a_1,\ldots,a_n\}]f^{\mathcal{T}}(\mathcal{C}) \\
(11) & f^{\mathcal{T}}([(\&_{i=1}^{n} a_i)\alpha]\mathcal{C}) & = & [\{a_1,\ldots,a_n\}]f^{\mathcal{T}}([\alpha]\mathcal{C}) \\
(12) & f^{\mathcal{T}}([\alpha+\beta]\mathcal{C}) & = & f^{\mathcal{T}}([\alpha]\mathcal{C}) \wedge f^{\mathcal{T}}([\beta]\mathcal{C}) \\
(13) & f^{\mathcal{T}}([\varphi?]\mathcal{C}) & = & f^{\mathcal{T}}(\varphi) \implies f^{\mathcal{T}}(\mathcal{C})
\end{array}
$$

$\mathcal{CL}$ Semantics

Example

- $f^{\mathcal{T}}(O(\&_{i=1}^{n} a_i)) = \langle\{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^{n} O_{a_i})$
  - "The Provider is obliged to provide internet and telephony services (at the same time)":

$$f^{\mathcal{T}}(O(a\&b)) = \langle\{a, b\}\rangle(O_a \wedge O_b)$$

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

$$f^{\mathcal{T}}(F(a)) = [a]\mathcal{F}_a$$

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

$$f^{\mathcal{T}}(P(a)) = \langle a\rangle\neg\mathcal{F}_a$$

$\mathcal{CL}$ Semantics

Example

- $f^{\mathcal{T}}(O(\&_{i=1}^{n} a_i)) = \langle\{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^{n} O_{a_i})$
  - "The Provider is obliged to provide internet and telephony services (at the same time)":

$$f^{\mathcal{T}}(O(a\&b)) = \langle\{a, b\}\rangle(O_a \wedge O_b)$$

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

$$f^{\mathcal{T}}(F(a)) = [a]\mathcal{F}_a$$

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

$$f^{\mathcal{T}}(P(a)) = \langle a\rangle\neg\mathcal{F}_a$$

$\mathcal{CL}$ Semantics

Example

- $f^{\mathcal{T}}(O(\&_{i=1}^{n} a_i)) = \langle\{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^{n} O_{a_i})$
  - "The Provider is obliged to provide internet and telephony services (at the same time)":

$$f^{\mathcal{T}}(O(a\&b)) = \langle\{a, b\}\rangle(O_a \wedge O_b)$$

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

$$f^{\mathcal{T}}(F(a)) = [a]\mathcal{F}_a$$

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

$$f^{\mathcal{T}}(P(a)) = \langle a\rangle\neg\mathcal{F}_a$$

$\mathcal{CL}$ Semantics

Example

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

  Applying the semantic encoding:

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

Example

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

  Applying the semantic encoding:

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

$\mathcal{CL}$ Semantics

Example

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

  Applying the semantic encoding:

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

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

  Applying the semantic encoding:

$$f^{\mathcal{T}}(F_{O(b)}(a)) = [a]\mathcal{F}_a \wedge [a]\langle b \rangle O_b$$

Plan

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

2. Properties of the Language

Properties of the contract language

Theorem

The following paradoxes are avoided in $\mathcal{CL}$:

- *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. $O(p \vee q)$

Problem

In SDL one can infer that $O(p) \Rightarrow O(p \vee q)$

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. $O(p \vee q)$

Problem

In SDL one can infer that $O(p) \Rightarrow O(p \vee q)$

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. $O(p \vee q)$

Problem

In SDL one can infer that $O(p) \Rightarrow O(p \vee q)$

Avoided in $\mathcal{CL}$

Proof Sketch:

- $f^{\mathcal{T}}(O(a)) = \langle a \rangle O_a$
- $O(a+b) \equiv O(a) \oplus O(b) \stackrel{f^{\mathcal{T}}}{=} \langle a \rangle O_a \wedge \langle b \rangle O_b$
- $\langle a \rangle O_a \not\Rightarrow \langle a \rangle O_a \wedge \langle b \rangle O_b$

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 $O(q) \wedge O(\neg q)$ (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 $O(q) \wedge O(\neg q)$ (due to 2)

Chisholm's Paradox (cont.)

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 $\mathcal{CL}$:

- $P(\alpha) \equiv \neg F(\alpha)$
- $O(\alpha) \Rightarrow P(\alpha)$
- $P(a) \not\Rightarrow P(a \& b)$
- $F(a) \not\Rightarrow F(a \& b)$
- $F(a \& b) \not\Rightarrow F(a)$
- $P(a \& b) \not\Rightarrow P(a)$

Final Remarks

We have seen...

- $\mathcal{CL}$: A formal language to write contracts
- The formal semantics given through an encoding into a μ-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 μ-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

- C. Prisacariu and G. Schneider. **A formal language for electronic contracts.** In FMOODS'07, vol. 4468 of LNCS, pp. 174-189, 2007