

# Specification and Analysis of Electronic Contracts

Gerardo Schneider  
(Joint work with Cristian Prisacariu and Gordon Pace)

Department of Informatics,  
University of Oslo

FLACOS'08  
Malta  
27-28 November 2008

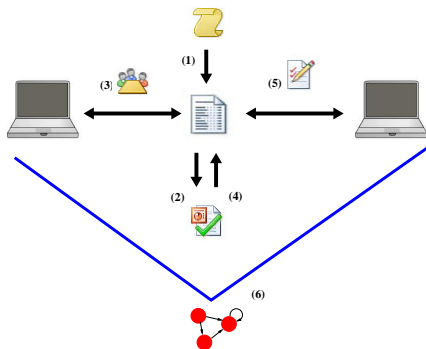
- “A **contract** is a binding agreement between two or more persons that is enforceable by law.” [Webster on-line]

- “A **contract** is a binding agreement between two or more persons that is enforceable by law.” [Webster on-line]
- ① Conventional contracts
  - Traditional commercial and judicial domain
- ② “Programming by contract” or “Design by contract” (e.g., Eiffel)
  - Pre- and post-conditions, invariants, temporal dependencies, etc
- ③ Behavioral interfaces
  - The allowed interactions are captured by legal (sets of) traces
- ④ In the context of web services (SOA)
  - Service-Level Agreement, an XML-like language (e.g. WSLA)
- ⑤ Contractual protocols
  - To specify the interaction between communicating entities
- ⑥ “Social contracts”: Multi-agent systems

- “A **contract** is a binding agreement between two or more persons that is enforceable by law.” [Webster on-line]
- ① Conventional contracts
  - Traditional commercial and judicial domain
- ② “Programming by contract” or “Design by contract” (e.g., Eiffel)
  - Pre- and post-conditions, invariants, temporal dependencies, etc
- ③ Behavioral interfaces
  - The allowed interactions are captured by legal (sets of) traces
- ④ In the context of web services (SOA)
  - Service-Level Agreement, an XML-like language (e.g. WSLA)
- ⑤ Contractual protocols
  - To specify the interaction between communicating entities
- ⑥ “Social contracts”: Multi-agent systems
- ⑦ “Deontic e-contracts”: representing Obligations, Permissions, Prohibitions

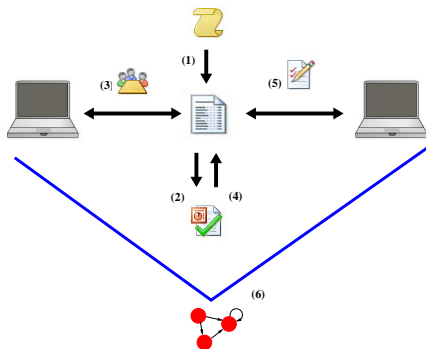
# Services and Contracts

- 1 Translate the informal contract into a formal language



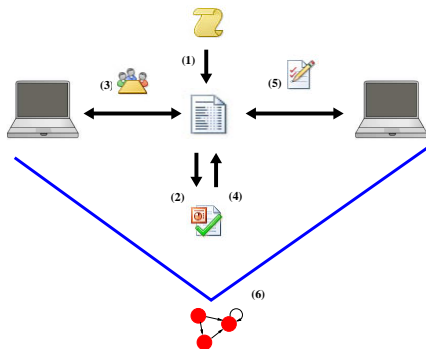
# Services and Contracts

- 1 Translate the informal contract into a formal language
- 2 Verify the contract (e.g., that it is contradiction-free)



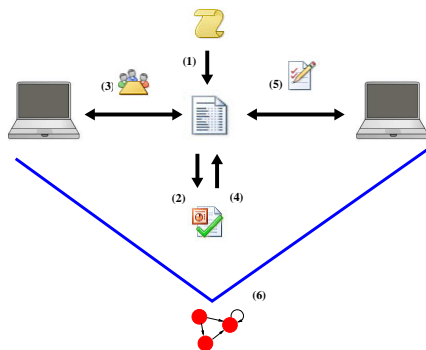
# Services and Contracts

- 1 Translate the informal contract into a formal language
- 2 Verify the contract (e.g., that it is contradiction-free)
- 3 Negotiate the contract



# Services and Contracts

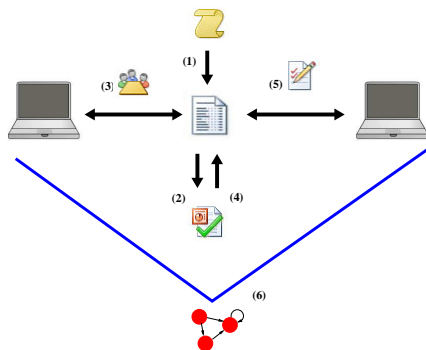
- 1 Translate the informal contract into a formal language
- 2 Verify the contract (e.g., that it is contradiction-free)
- 3 Negotiate the contract
- 4 After negotiation verify the contract again





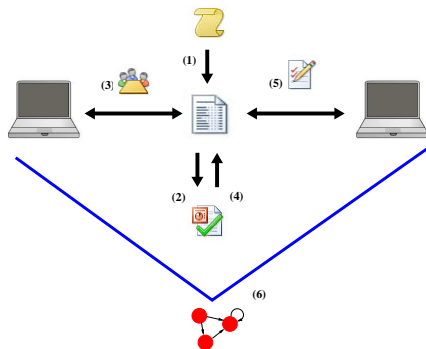
# Services and Contracts

- 1 Translate the informal contract into a formal language
- 2 Verify the contract (e.g., that it is contradiction-free)
- 3 Negotiate the contract
- 4 After negotiation verify the contract again
- 5 Obtain the final contract and “sign” it



# Services and Contracts

- 1 Translate the informal contract into a formal language
- 2 Verify the contract (e.g., that it is contradiction-free)
- 3 Negotiate the contract
- 4 After negotiation verify the contract again
- 5 Obtain the final contract and “sign” it
- 6 Monitor/enforce contract fulfillment



- Use **deontic e-contracts** to 'rule' services exchange
- 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 Develop a **theory of contracts**
  - Contract composition
  - Subcontracting
  - Conformance between a contract and the governing policies
  - *Meta-contracts* (policies)
- ④ **Monitor** contracts
  - Run-time system to ensure the contract is respected
  - In case of contract violations, act accordingly

- Use **deontic e-contracts** to 'rule' services exchange
- 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 Develop a **theory of contracts**
  - Contract composition
  - Subcontracting
  - Conformance between a contract and the governing policies
  - *Meta-contracts* (policies)
- ④ **Monitor** contracts
  - Run-time system to ensure the contract is respected
  - In case of contract violations, act accordingly

- Use **deontic e-contracts** to 'rule' services exchange
- 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 Develop a **theory of contracts**
  - Contract composition
  - Subcontracting
  - Conformance between a contract and the governing policies
  - *Meta-contracts* (policies)
- ④ **Monitor** contracts
  - Run-time system to ensure the contract is respected
  - In case of contract violations, act accordingly

- Use **deontic e-contracts** to 'rule' services exchange
- 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 Develop a **theory of contracts**
  - Contract composition
  - Subcontracting
  - Conformance between a contract and the governing policies
  - *Meta-contracts* (policies)
- ④ **Monitor** contracts
  - Run-time system to ensure the contract is respected
  - In case of contract violations, act accordingly

- Use **deontic e-contracts** to 'rule' services exchange
- 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 Develop a **theory of contracts**
  - Contract composition
  - Subcontracting
  - Conformance between a contract and the governing policies
  - *Meta-contracts* (policies)
- ④ **Monitor** contracts
  - Run-time system to ensure the contract is respected
  - In case of contract violations, act accordingly

- 1 The Contract Language  $\mathcal{CL}$
- 2 Model Checking Contracts
- 3 Final Remarks



# The Contract Specification Language $\mathcal{CL}$

## Definition ( $\mathcal{CL}$ Syntax)

$$\begin{aligned} \text{Contract} &:= \mathcal{D} ; \mathcal{C} \\ \mathcal{C} &:= \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle \alpha \rangle \mathcal{C} \mid \mathcal{C} \mathcal{U} \mathcal{C} \mid \bigcirc \mathcal{C} \mid \square \mathcal{C} \\ \mathcal{C}_O &:= O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\ \mathcal{C}_P &:= P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\ \mathcal{C}_F &:= F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F \end{aligned}$$

- $O(\alpha)$ ,  $P(\alpha)$ ,  $F(\alpha)$ : obligations, permissions, and prohibitions
- $\alpha$  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  $\square$  correspond to **temporal logic operators**

## Definition ( $\mathcal{CL}$ Syntax)

$$\begin{aligned} \text{Contract} &:= \mathcal{D} ; \mathcal{C} \\ \mathcal{C} &:= \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle \alpha \rangle \mathcal{C} \mid \mathcal{C} \mathcal{U} \mathcal{C} \mid \bigcirc \mathcal{C} \mid \square \mathcal{C} \\ \mathcal{C}_O &:= O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\ \mathcal{C}_P &:= P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\ \mathcal{C}_F &:= F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F \end{aligned}$$

- $O(\alpha)$ ,  $P(\alpha)$ ,  $F(\alpha)$ : obligations, permissions, and prohibitions
- $\alpha$  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  $\square$  correspond to **temporal logic operators**

## Definition ( $\mathcal{CL}$ Syntax)

$$\begin{aligned} \text{Contract} &:= \mathcal{D} ; \mathcal{C} \\ \mathcal{C} &:= \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle \alpha \rangle \mathcal{C} \mid \mathcal{C} \mathcal{U} \mathcal{C} \mid \bigcirc \mathcal{C} \mid \square \mathcal{C} \\ \mathcal{C}_O &:= O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\ \mathcal{C}_P &:= P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\ \mathcal{C}_F &:= F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F \end{aligned}$$

- $O(\alpha)$ ,  $P(\alpha)$ ,  $F(\alpha)$ : obligations, permissions, and prohibitions
- $\alpha$  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  $\square$  correspond to **temporal logic operators**

## Definition ( $\mathcal{CL}$ Syntax)

$$\begin{aligned} \text{Contract} &:= \mathcal{D} ; \mathcal{C} \\ \mathcal{C} &:= \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle \alpha \rangle \mathcal{C} \mid \mathcal{C} \mathcal{U} \mathcal{C} \mid \bigcirc \mathcal{C} \mid \square \mathcal{C} \\ \mathcal{C}_O &:= O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\ \mathcal{C}_P &:= P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\ \mathcal{C}_F &:= F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F \end{aligned}$$

- $O(\alpha)$ ,  $P(\alpha)$ ,  $F(\alpha)$ : obligations, permissions, and prohibitions
- $\alpha$  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  $\square$  correspond to **temporal logic operators**

## Definition ( $\mathcal{CL}$ Syntax)

$$\begin{aligned} \text{Contract} &:= \mathcal{D} ; \mathcal{C} \\ \mathcal{C} &:= \mathcal{C}_O \mid \mathcal{C}_P \mid \mathcal{C}_F \mid \mathcal{C} \wedge \mathcal{C} \mid [\alpha]\mathcal{C} \mid \langle \alpha \rangle \mathcal{C} \mid \mathcal{C} \mathcal{U} \mathcal{C} \mid \bigcirc \mathcal{C} \mid \square \mathcal{C} \\ \mathcal{C}_O &:= O(\alpha) \mid \mathcal{C}_O \oplus \mathcal{C}_O \\ \mathcal{C}_P &:= P(\alpha) \mid \mathcal{C}_P \oplus \mathcal{C}_P \\ \mathcal{C}_F &:= F(\alpha) \mid \mathcal{C}_F \vee [\alpha]\mathcal{C}_F \end{aligned}$$

- $O(\alpha)$ ,  $P(\alpha)$ ,  $F(\alpha)$ : obligations, permissions, and prohibitions
- $\alpha$  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  $\square$  correspond to **temporal logic operators**

# More on the Contract Language

## CTD and CTP

- We want to handle **violations** (*CTDs*, *CTPs*)
  - A **contrary-to-duty** (CTD) expresses what happen when an obligation is not fulfilled
  - A **contrary-to-prohibition** (CTP) defines what is to be done when a prohibition is violated

# More on the Contract Language

## CTD and CTP

- We want to handle **violations** (*CTDs*, *CTPs*)
  - A **contrary-to-duty** (CTD) expresses what happens when an obligation is not fulfilled
  - A **contrary-to-prohibition** (CTP) defines what is to be done when a prohibition is violated

### Example

**CTD:** You must send an acknowledgment immediately after receiving the message. If you don't do that, you must pay double.

**CTP:** You are forbidden to send a message before having acknowledged the reception of the previous answer. If you do that, I am allowed to cancel the contract.

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

$$O_C(\alpha) = O(\alpha) \wedge [\bar{\alpha}]C$$



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

$$O_C(\alpha) = O(\alpha) \wedge [\bar{\alpha}]C$$

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

$$F_C(\alpha) = F(\alpha) \wedge [\alpha]C$$

# $\mathcal{CL}$ Semantics

$\mathcal{C}\mu$  – A variant of the modal  $\mu$ -calculus

- Translation into a variant of  $\mu$ -calculus ( $\mathcal{C}\mu$ )
- The syntax of the  $\mathcal{C}\mu$  logic

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

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

- 1  $P_c$  is set of propositional constants  $O_a$  and  $\mathcal{F}_a$ , one for each basic action  $a$
- 2 **Multisets of basic actions:** i.e.  $\gamma = \{a, a, b\}$  is a label

- Translation into a variant of  $\mu$ -calculus ( $\mathcal{C}\mu$ )
- The syntax of the  $\mathcal{C}\mu$  logic
$$\varphi := P \mid Z \mid P_c \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\gamma]\varphi \mid \mu Z.\varphi(Z)$$

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

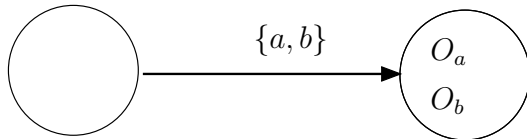
- 1  $P_c$  is set of propositional constants  $O_a$  and  $\mathcal{F}_a$ , one for each basic action  $a$
- 2 **Multisets of basic actions:** i.e.  $\gamma = \{a, a, b\}$  is a label

- Obligation

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

- Obligation

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



$O(a\&b)$

# Model Checking Contracts

- 1 Model the conventional contract (in English) as a  $\mathcal{CL}$  expression
- 2 Translate the  $\mathcal{CL}$  specification into  $\mathcal{C}\mu$
- 3 Obtain a Kripke-like model (LTS) from the  $\mathcal{C}\mu$  formulas
- 4 Translate the LTS into the input language of NuSMV
- 5 Perform model checking using NuSMV
  - Check the model is 'good'
  - Check some properties about the client and the provider
- 6 In case of a counter-example given by NuSMV, interpret it as a  $\mathcal{CL}$  clause and repeat the model checking process until the property is satisfied
- 7 In some cases rephrase the original contract

# Model Checking Contracts

- 1 Model the conventional contract (in English) as a  $\mathcal{CL}$  expression
- 2 Translate the  $\mathcal{CL}$  specification into  $\mathcal{C}\mu$
- 3 Obtain a Kripke-like model (LTS) from the  $\mathcal{C}\mu$  formulas
- 4 Translate the LTS into the input language of NuSMV
- 5 Perform model checking using NuSMV
  - Check the model is 'good'
  - Check some properties about the client and the provider
- 6 In case of a counter-example given by NuSMV, interpret it as a  $\mathcal{CL}$  clause and repeat the model checking process until the property is satisfied
- 7 In some cases rephrase the original contract

# Model Checking Contracts

- 1 Model the conventional contract (in English) as a  $\mathcal{CL}$  expression
- 2 Translate the  $\mathcal{CL}$  specification into  $\mathcal{C}\mu$
- 3 Obtain a Kripke-like model (LTS) from the  $\mathcal{C}\mu$  formulas
- 4 Translate the LTS into the input language of NuSMV
- 5 Perform model checking using NuSMV
  - Check the model is 'good'
  - Check some properties about the client and the provider
- 6 In case of a counter-example given by NuSMV, interpret it as a  $\mathcal{CL}$  clause and repeat the model checking process until the property is satisfied
- 7 In some cases rephrase the original contract



# Model Checking Contracts

- 1 Model the conventional contract (in English) as a  $\mathcal{CL}$  expression
- 2 Translate the  $\mathcal{CL}$  specification into  $\mathcal{C}\mu$
- 3 Obtain a Kripke-like model (LTS) from the  $\mathcal{C}\mu$  formulas
- 4 Translate the LTS into the input language of NuSMV
- 5 Perform model checking using NuSMV
  - Check the model is 'good'
  - Check some properties about the client and the provider
- 6 In case of a counter-example given by NuSMV, interpret it as a  $\mathcal{CL}$  clause and repeat the model checking process until the property is satisfied
- 7 In some cases rephrase the original contract

# Model Checking Contracts

- 1 Model the conventional contract (in English) as a  $\mathcal{CL}$  expression
- 2 Translate the  $\mathcal{CL}$  specification into  $\mathcal{C}\mu$
- 3 Obtain a Kripke-like model (LTS) from the  $\mathcal{C}\mu$  formulas
- 4 Translate the LTS into the input language of NuSMV
- 5 Perform model checking using NuSMV
  - Check the model is 'good'
  - Check some properties about the client and the provider
- 6 In case of a counter-example given by NuSMV, interpret it as a  $\mathcal{CL}$  clause and repeat the model checking process until the property is satisfied
- 7 In some cases rephrase the original contract

# Model Checking Contracts

- 1 Model the conventional contract (in English) as a  $\mathcal{CL}$  expression
- 2 Translate the  $\mathcal{CL}$  specification into  $\mathcal{C}\mu$
- 3 Obtain a Kripke-like model (LTS) from the  $\mathcal{C}\mu$  formulas
- 4 Translate the LTS into the input language of NuSMV
- 5 Perform model checking using NuSMV
  - Check the model is 'good'
  - Check some properties about the client and the provider
- 6 In case of a counter-example given by NuSMV, interpret it as a  $\mathcal{CL}$  clause and repeat the model checking process until the property is satisfied
- 7 In some cases rephrase the original contract

# Model Checking Contracts

- 1 Model the conventional contract (in English) as a  $\mathcal{CL}$  expression
- 2 Translate the  $\mathcal{CL}$  specification into  $\mathcal{C}\mu$
- 3 Obtain a Kripke-like model (LTS) from the  $\mathcal{C}\mu$  formulas
- 4 Translate the LTS into the input language of NuSMV
- 5 Perform model checking using NuSMV
  - Check the model is 'good'
  - Check some properties about the client and the provider
- 6 In case of a counter-example given by NuSMV, interpret it as a  $\mathcal{CL}$  clause and repeat the model checking process until the property is satisfied
- 7 In some cases rephrase the original contract

# Case Study

## A Contract Example

1. The **Client** shall not:
  - a) supply false information to the Client Relations Department of the **Provider**.
2. Whenever the Internet Traffic is **high** then the **Client** must pay [*price*] immediately, or the **Client** must notify the **Provider** by sending an e-mail specifying that he will pay later.
3. If the **Client** delays the payment as stipulated in 2, after notification he must immediately lower the Internet traffic to the **normal** level, and pay later twice ( $2 * [price]$ ).
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay  $3 * [price]$ .
5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider's** web page to the Client Relations Department of the **Provider**.
6. **Provider** may, at its sole discretion, without notice or giving any reason or incurring any liability for doing so:
  - a) Suspend Internet Services immediately if **Client** is in breach of Clause 1;

# Case Study

## A Contract Example

1. The **Client** shall not:

a) supply false information to the Client Relations Department of the **Provider**.

2. Whenever the Internet Traffic is **high** then the **Client** must pay [*price*] immediately, or the **Client** must notify the **Provider** by sending an e-mail specifying that he will pay later.

3. If the **Client** delays the payment as stipulated in 2, after notification he must immediately lower the Internet traffic to the **normal** level, and pay later twice ( $2 * [price]$ ).

4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay  $3 * [price]$ .

5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider's** web page to the Client Relations Department of the **Provider**.

6. **Provider** may, at its sole discretion, without notice or giving any reason or incurring any liability for doing so:

a) Suspend Internet Services immediately if **Client** is in breach of Clause 1;

# Case Study

## Translating into $\mathcal{CL}$ syntax

1.  $\square F(fi)$
2. Whenever the Internet Traffic is **high** then the **Client** must pay [*price*] immediately, or the **Client** must notify the **Provider** by sending an e-mail specifying that he will pay later.
3. If the **Client** delays the payment as stipulated in 2, after notification he must immediately lower the Internet traffic to the **normal** level, and pay later twice ( $2 * [price]$ ).
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay  $3 * [price]$ .
5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider's** web page to the Client Relations Department of the **Provider**.
6. **Provider** may, at its sole discretion, without notice or giving any reason or incurring any liability for doing so:
  - a) Suspend Internet Services immediately if **Client** is in breach of Clause 1;

# Case Study

## Translating into $\mathcal{CL}$ syntax

1.  $\square F(fi)$
2. Whenever the Internet Traffic is **high** then the **Client** must pay [*price*] immediately, or the **Client** must notify the **Provider** by sending an e-mail specifying that he will pay later.
3. If the **Client** delays the payment as stipulated in 2, after notification he must immediately lower the Internet traffic to the **normal** level, and pay later twice ( $2 * [price]$ ).
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay  $3 * [price]$ .
5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider's** web page to the Client Relations Department of the **Provider**.
6. **Provider** may, at its sole discretion, without notice or giving any reason or incurring any liability for doing so:
  - a) Suspend Internet Services immediately if **Client** is in breach of Clause 1;



# Case Study

## Translating into $\mathcal{CL}$ syntax

1.  $\square F_{P(s)}(fi)$
2. Whenever the Internet Traffic is **high** then the **Client** must pay [*price*] immediately, or the **Client** must notify the **Provider** by sending an e-mail specifying that he will pay later.
3. If the **Client** delays the payment as stipulated in 2, after notification he must immediately lower the Internet traffic to the **normal** level, and pay later twice ( $2 * [price]$ ).
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay  $3 * [price]$ .
5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider's** web page to the Client Relations Department of the **Provider**.

# Case Study

## Translating into $\mathcal{CL}$ syntax

1.  $\Box F_{P(s)}(fi)$
2.  $\Box[h](\phi \Rightarrow O(p + (d\&n)))$
3. If the **Client** delays the payment as stipulated in 2, after notification he must immediately lower the Internet traffic to the **normal** level, and pay later twice ( $2 * [price]$ ).
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay  $3 * [price]$ .
5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider's** web page to the Client Relations Department of the **Provider**.

# Case Study

## Translating into $\mathcal{CL}$ syntax

1.  $\Box F_{P(s)}(fi)$
2.  $\Box[h](\phi \Rightarrow O(p + (d\&n)))$
3.  $\Box([d\&n](O(l) \wedge [l]\Diamond O(p\&p)))$
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay  $3 * [price]$ .
5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider's** web page to the Client Relations Department of the **Provider**.

# Case Study

## Translating into $\mathcal{CL}$ syntax

1.  $\Box F_{P(s)}(fi)$
2.  $\Box[h](\phi \Rightarrow O(p + (d\&n)))$
3.  $\Box([d\&n](O(l) \wedge [l]\Diamond O(p\&p)))$
4.  $\Box([d\&n \cdot \bar{l}]\Diamond O(p\&p\&p))$
5. The **Client** shall, as soon as the Internet Service becomes operative, submit within seven (7) days the Personal Data Form from his account on the **Provider's** web page to the Client Relations Department of the **Provider**.

# Case Study

## Translating into $\mathcal{CL}$ syntax

1.  $\Box F_{P(s)}(fi)$
2.  $\Box[h](\phi \Rightarrow O(p + (d\&n)))$
3.  $\Box([d\&n](O(l) \wedge [l]\Diamond O(p\&p)))$
4.  $\Box([d\&n \cdot \bar{l}]\Diamond O(p\&p\&p))$
5.  $\Box([o]O(sfD))$

# Case Study

## Handcrafting the model

$\phi$  = the Internet traffic is high

$fi$  = client supplies false information  
to Client Relations Department

$h$  = client increases Internet traffic  
to *high* level

$p$  = client pays [price]

$d$  = client delays payment

$n$  = client notifies by e-mail

$l$  = client lowers the Int. traffic

$sfD$  = client sends the Personal  
Data Form to Client Relations  
Department

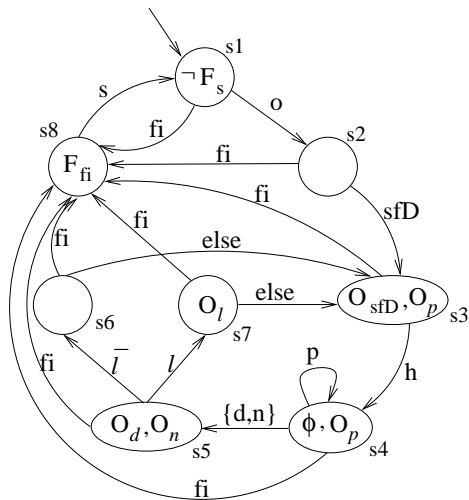
$o$  = provider activates the Internet  
Service (it becomes operative)

$s$  = provider suspends service

# Case Study

## Handcrafting the model

- $\phi$  = the Internet traffic is high
- $fi$  = client supplies false information to Client Relations Department
- $h$  = client increases Internet traffic to *high* level
- $p$  = client pays [price]
- $d$  = client delays payment
- $n$  = client notifies by e-mail
- $l$  = client lowers the Int. traffic
- $sfD$  = client sends the Personal Data Form to Client Relations Department
- $o$  = provider activates the Internet Service (it becomes operative)
- $s$  = provider suspends service



Use of model checking for reasoning about contracts:

- 1 We use model checking to increase our confidence in the correctness of the model with respect to the original natural language contract
  - 2 By finding errors in the model, we identify problems in the original natural language contract or its interpretation in  $\mathcal{CL}$
  - 3 We enable the signatories to safeguard their interests by ensuring certain desirable properties hold (and certain undesirable ones do not)
- Counter-examples
    - Problems on the  $\mathcal{CL}$  formula and on the original contract in English



# Final Remarks

- A **formal specification language for contracts** with semantics based on a variant of  $\mu$ -calculus
- Initial ideas on how to model check contracts

- A **formal specification language for contracts** with semantics based on a variant of  $\mu$ -calculus
- Initial ideas on how to model check contracts

Currently:

- Redesign  $\mathcal{CL}$
- Kripke semantics for  $\mathcal{CL}$ 
  - Development of an action algebra
- Automatic monitor extraction

- Develop a proof system for (an improved)  $\mathcal{CL}$
- Internal vs external operations
- Add time
- Automate the model checking process
- Develop a theory of contracts
- Programming languages and contracts
  - Embedded language
  - Contract-as-types
- Combination with operational models (e.g. process algebra)
- Case studies and other applications:
  - Fault tolerant systems
  - Long transactions
  - Component-based development
  - Legal contracts (?)

- Develop a proof system for (an improved)  $\mathcal{CL}$
- Internal vs external operations
- Add time
- Automate the model checking process
- Develop a theory of contracts
- Programming languages and contracts
  - Embedded language
  - Contract-as-types
- Combination with operational models (e.g. process algebra)
- Case studies and other applications:
  - Fault tolerant systems
  - Long transactions
  - Component-based development
  - Legal contracts (?)

- Develop a proof system for (an improved)  $\mathcal{CL}$
- Internal vs external operations
- Add time
- Automate the model checking process
- Develop a theory of contracts
- Programming languages and contracts
  - Embedded language
  - Contract-as-types
- Combination with operational models (e.g. process algebra)
- Case studies and other applications:
  - Fault tolerant systems
  - Long transactions
  - Component-based development
  - Legal contracts (?)

- Develop a proof system for (an improved)  $\mathcal{CL}$
- Internal vs external operations
- Add time
- Automate the model checking process
- Develop a theory of contracts
- Programming languages and contracts
  - Embedded language
  - Contract-as-types
- Combination with operational models (e.g. process algebra)
- Case studies and other applications:
  - Fault tolerant systems
  - Long transactions
  - Component-based development
  - Legal contracts (?)

- Develop a proof system for (an improved)  $\mathcal{CL}$
- Internal vs external operations
- Add time
- Automate the model checking process
- Develop a theory of contracts
- Programming languages and contracts
  - Embedded language
  - Contract-as-types
- Combination with operational models (e.g. process algebra)
- Case studies and other applications:
  - Fault tolerant systems
  - Long transactions
  - Component-based development
  - Legal contracts (?)

- Develop a proof system for (an improved)  $\mathcal{CL}$
- Internal vs external operations
- Add time
- Automate the model checking process
- Develop a theory of contracts
- Programming languages and contracts
  - Embedded language
  - Contract-as-types
- Combination with operational models (e.g. process algebra)
- Case studies and other applications:
  - Fault tolerant systems
  - Long transactions
  - Component-based development
  - Legal contracts (?)



- Develop a proof system for (an improved)  $\mathcal{CL}$
- Internal vs external operations
- Add time
- Automate the model checking process
- Develop a theory of contracts
- Programming languages and contracts
  - Embedded language
  - Contract-as-types
- Combination with operational models (e.g. process algebra)
- Case studies and other applications:
  - Fault tolerant systems
  - Long transactions
  - Component-based development
  - Legal contracts (?)

- Develop a proof system for (an improved)  $\mathcal{CL}$
- Internal vs external operations
- Add time
- Automate the model checking process
- Develop a theory of contracts
- Programming languages and contracts
  - Embedded language
  - Contract-as-types
- Combination with operational models (e.g. process algebra)
- Case studies and other applications:
  - Fault tolerant systems
  - Long transactions
  - Component-based development
  - Legal contracts (?)

# What We Have Done So Far...

- C. Prisacariu and G. Schneider. **A formal language for electronic contracts.** In FMOODS'07, LNCS.
- G. Pace, C. Prisacariu and G. Schneider. **Model Checking Contracts –A case study.** In ATVA'07, LNCS.
- M. Kyas, C. Prisacariu, and G. Schneider. **Runtime Monitoring of Electronic Contracts.** In ATVA'08, LNCS.
- G. Pace and G. Schneider. **Challenges in the specification of full contracts.** In iFM'09, to appear in LNCS.

# What We Have Done So Far...

- C. Prisacariu and G. Schneider. **A formal language for electronic contracts.** In FMOODS'07, LNCS.
- G. Pace, C. Prisacariu and G. Schneider. **Model Checking Contracts –A case study.** In ATVA'07, LNCS.
- M. Kyas, C. Prisacariu, and G. Schneider. **Runtime Monitoring of Electronic Contracts.** In ATVA'08, LNCS.
- G. Pace and G. Schneider. **Challenges in the specification of full contracts.** In iFM'09, to appear in LNCS.

Tomorrow Gordon will present part of Stephen's master thesis:

- S. Fenech. **Conflict analysis of deontic contracts.** M.Sc. thesis. University of Malta, November 2008.