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]
Contracts and Informatics

“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
1. Translate the informal contract into a formal language
1. Translate the informal contract into a formal language

2. Verify the contract (e.g., that it is contradiction-free)
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
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

Gerardo Schneider (UiO)  Specification and Analysis of Contracts  FLACOS’08, Malta 3 / 18
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
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
Aim and Motivation

- 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
Aim and Motivation

- 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
Aim and Motivation

- 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
Aim and Motivation

- 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
Aim and Motivation

- 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
Outline

1. The Contract Language $CL$

2. Model Checking Contracts

3. Final Remarks
### Definition (CL Syntax)

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

- \(O(\alpha), P(\alpha), F(\alpha)\): obligations, permissions, and prohibitions
- \(\alpha\) 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 \(\Box\) correspond to temporal logic operators
The Contract Specification Language $\mathcal{CL}$

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

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

- $O(\alpha)$, $P(\alpha)$, $F(\alpha)$: obligations, permissions, and prohibitions
- $\alpha$: actions given in the definition part $D$
  - + choice
  - 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
- $\mathcal{U}$, $\bigodot$, and $\square$ correspond to temporal logic operators
The Contract Specification Language \( \mathcal{CL} \)

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

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

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

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

\[
\begin{align*}
\text{Contract} & := D \ ; \ C \\
C & := C_O \mid C_P \mid C_F \mid C \land C \mid [\alpha]C \mid \langle \alpha \rangle C \mid C U C \mid \bigcirc C \mid \square C \\
C_O & := O(\alpha) \mid C_O \oplus C_O \\
C_P & := P(\alpha) \mid C_P \oplus C_P \\
C_F & := F(\alpha) \mid C_F \lor [\alpha]C_F
\end{align*}
\]

- $O(\alpha), P(\alpha), F(\alpha)$: obligations, permissions, and prohibitions
- $\alpha$ 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
- $\mathcal{U}, \bigcirc, \text{and } \square$ correspond to temporal logic operators

Gerardo Schneider (UiO) Specification and Analysis of Contracts FLACOS’08, Malta 6 / 18
The Contract Specification Language $\mathcal{CL}$

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

$$
\text{Contract} := \mathcal{D} ; \mathcal{C} \\
\mathcal{C} := \mathcal{C}_O | \mathcal{C}_P | \mathcal{C}_F | \mathcal{C} \land \mathcal{C} | [\alpha]\mathcal{C} | \langle\alpha\rangle\mathcal{C} | \mathcal{C} \cup \mathcal{C} | \Diamond \mathcal{C} | \Box \mathcal{C} \\
\mathcal{C}_O := O(\alpha) | \mathcal{C}_O \oplus \mathcal{C}_O \\
\mathcal{C}_P := P(\alpha) | \mathcal{C}_P \oplus \mathcal{C}_P \\
\mathcal{C}_F := F(\alpha) | \mathcal{C}_F \lor [\alpha]\mathcal{C}_F
$$

- $O(\alpha)$, $P(\alpha)$, $F(\alpha)$: obligations, permissions, and prohibitions
- $\alpha$ are actions given in the definition part $\mathcal{D}$
  - + choice
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrency
  - $\phi?$ test
- $\land$, $\lor$, 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
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 happen 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.
More on the Contract Language
CTD and CTP

Expressing contrary-to-duty (CTD)

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

\[ F_C(\alpha) = F(\alpha) \land [\alpha]C \]
Translation into a variant of $\mu$-calculus ($C\mu$)

The syntax of the $C\mu$ logic

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

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

1. $P_c$ is set of propositional constants $O_a$ and $F_a$, 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 ($C\mu$)

The syntax of the $C\mu$ logic

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

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

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

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

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

\[
\begin{array}{c}
\xrightarrow{\{a, b\}} \\
O(a \& b)
\end{array}
\]

\[
\begin{array}{c}
\xrightarrow{O_a}
\xrightarrow{O_b}
\end{array}
\]
Model Checking Contracts

1. Model the conventional contract (in English) as a $CL$ expression
2. Translate the $CL$ specification into $C\mu$
3. Obtain a Kripke-like model (LTS) from the $C\mu$ 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 $CL$ 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 $CL$ expression
2. Translate the $CL$ specification into $C\mu$
3. Obtain a Kripke-like model (LTS) from the $C\mu$ 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 $CL$ 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 $CL$ expression
2. Translate the $CL$ specification into $C_{\mu}$
3. Obtain a Kripke-like model (LTS) from the $C_{\mu}$ 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 $CL$ 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 $C_\mu$
3. Obtain a Kripke-like model (LTS) from the $C_\mu$ 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 $C_\mu$
3. Obtain a Kripke-like model (LTS) from the $C_\mu$ 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
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 \((2 \times [price])\).
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay \(3 \times [price]\).
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;
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 \((2 \times [price])\).
4. If the **Client** does not lower the Internet traffic immediately, then the **Client** will have to pay \(3 \times [price]\).
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;
1. □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 \times \[price\])\).

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

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;
1. $\Box F(fi)$

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 ($2 \times [price]$).

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

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;
1. $\Box F_{P(s)}(fi)$

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 ($2 \times [price]$).

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

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 $(2 \ast [price])$.

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

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 $CL$ syntax

1. $\Box F_{P(s)}(fi)$

2. $\Box[h](\phi \Rightarrow O(p + (d & n)))$

3. $\Box([d & n](O(l) \land [l] O(p & p)))$

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

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. $\Box([d\&n](O(l) \land [l]\Diamond O(p\&p)))$

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. $\square F_{P(s)}(fi)$

2. $\square [h](\phi \Rightarrow O(p + (d\&n)))$

3. $\square ([d\&n](O(l) \land [l]\diamond O(p\&p)))$

4. $\square ([d\&n \cdot \bar{l}]\diamond O(p\&p\&p))$

5. $\square ([o] O(sfD))$
Case Study
Handcrafting the model

φ = 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
\( f_i = \) 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

Gerardo Schneider (UiO) Specification and Analysis of Contracts FLACOS’08, Malta 14 / 18
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 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 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
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

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

- 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 (?)
Future Work

- Develop a proof system for (an improved) $CL$
- 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 (?)
Future Work

- 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 (?)
Future Work

- Develop a proof system for (an improved) $CL$
- 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 (?)
Future Work

- Develop a proof system for (an improved) $CL$
- 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 (?)
Future Work

- 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 (?)
Future Work

- 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 (?)
Future Work

- Develop a proof system for (an improved) $CL$
- 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.
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: