On the Specification of Full Contracts

Stephen Fenech\textsuperscript{1}  Joseph Okika\textsuperscript{2}  Gordon Pace\textsuperscript{1}  Anders Ravn\textsuperscript{2}  Gerardo Schneider\textsuperscript{3}

sfen002@um.edu.mt  ojc@cs.aau.dk  gordon.pace@um.edu.mt  apr@cs.aau.dk  gerardo@ifi.uio.no

\textsuperscript{1}Dept. of Computer Science – University of Malta, Malta

\textsuperscript{2}Dept. of Computer Science – Aalborg University, Denmark

\textsuperscript{3}Dept. of Informatics – University of Oslo, Norway

FESCA, 28 March 2009 – York, UK
A **contract** is a binding agreement between two or more entities (enforceable by law)

- Use contracts to regulate interactions in concurrent/distributed systems
  - Components, services, etc
- Different notions (or *levels*) of contracts
  - Static interfaces
  - Behavioural interfaces
  - *Design-by-contract* (pre-, post-conditions, invariants, etc)
  - Quality-of-Service
  - ‘Social’ contracts
  - Deontic e-contracts
A contract is a binding agreement between two or more entities (enforceable by law)

Use contracts to regulate interactions in concurrent/distributed systems

Components, services, etc

Different notions (or levels) of contracts

Static interfaces

Behavioural interfaces

Design-by-contract (pre-, post-conditions, invariants, etc)

Quality-of-Service

‘Social’ contracts

Deontic e-contracts
A contract is a binding agreement between two or more entities (enforceable by law)

Use contracts to regulate interactions in concurrent/distributed systems
  - Components, services, etc

Different notions (or levels) of contracts
  - Static interfaces
  - Behavioural interfaces
  - Design-by-contract (pre-, post-conditions, invariants, etc)
  - Quality-of-Service
  - ‘Social’ contracts
  - Deontic e-contracts
A contract is a binding agreement between two or more entities (enforceable by law)

Use contracts to regulate interactions in concurrent/distributed systems
- Components, services, etc

Different notions (or levels) of contracts
- Static interfaces
- Behavioural interfaces
- Design-by-contract (pre-, post-conditions, invariants, etc)
- Quality-of-Service
- ‘Social’ contracts
- Deontic e-contracts

Full contracts: Normal and exceptional behaviour
- Exceptions, compensations, tolerance to faults, penalties, etc
Objectives of Our Work

1. Specification of CoCoME (use cases 1-8) using \( CL \)
   - \( CL \) is contract language based on deontic logic
   - It allows the specification of obligations, permissions and prohibitions, and the penalties in case of violations

2. To compare suitability of operational and logical approaches to specify full contracts on a well-known case study (CoCoME)
   - Operational
     - rCOS (Relational Calculus of Object and Component Systems –CSP implementation)
   - Logical
     - Deontic-logic based language (\( CL \))
     - Temporal logics
CoCoME: Common Component Modelling Example

- Trading System to handle sales and inventory of a store chain
- 8 use cases
CoCoME: Common Component Modelling Example

- Trading System to handle sales and inventory of a store chain
- 8 use cases
  - **Use case 1**
    - How a sale is processed
  - **Use case 2**
    - How a cash desk switches to express mode, restricting total number of customer items
- We focus on the behavioural aspects of the use cases
  - Prop1, Prop2, Prop3
Definition (CSP)

CSP (rCOS)

\[ P ::= \text{Stop} \mid a \rightarrow P \mid P[]P \mid P \land P \mid X \]
Operational and Logic Specification Languages
CSP and Temporal Logics

**Definition (CSP)**

- **CSP** (rCOS)

\[
P ::= \text{Stop} \mid a \rightarrow P \mid P \parallel P \mid P \mid X
\]

**Definition (Temporal Logics)**

- **LTL**

\[
\varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid G\varphi \mid F\varphi \mid X\varphi \mid \varphi U \varphi
\]

- **CTL**

\[
\varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid AG\varphi \mid AF\varphi \mid AX\varphi \mid \varphi AU \varphi \mid EG\varphi \mid EF\varphi \mid EX\varphi \mid \varphi EU \varphi
\]
Definition ($CL$ Syntax)

\[
\begin{align*}
C & := C_O | C_P | C_F | C \land C | [\beta]C | \top | \bot \\
C_O & := \Box_C(\alpha) | C_O \oplus C_O \\
C_P & := \Box_P(\alpha) | C_P \oplus C_P \\
C_F & := \Box_F(\alpha) | C_F \lor [\alpha]C_F \\
\alpha & := 0 | 1 | a | \alpha \land \alpha | \alpha \mid \alpha + \alpha \\
\beta & := 0 | 1 | a | \beta \land \beta | \beta ; \beta | \beta + \beta | \beta^* 
\end{align*}
\]
Operational and Logic Specification Languages

$CL$: A Deontic-Based Language for Contracts

**Definition ($CL$ Syntax)**

\[C := C_O \mid C_P \mid C_F \mid C \land C \mid [\beta]C \mid \top \mid \bot\]

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

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

\[C_F := \mathcal{F}_C(\alpha) \mid C_F \lor [\alpha]C_F\]

\[\alpha := 0 \mid 1 \mid a \mid \alpha \& \alpha \mid \alpha \mid \alpha + \alpha\]

\[\beta := 0 \mid 1 \mid a \mid \beta \& \beta \mid \beta \mid \beta + \beta \mid \beta^*\]

**Example (Specification of Prop1)**

If pay with card, three allowed attempts to enter pin code; otherwise pay with cash, or return goods

\[\square[\text{cardPay}] \quad \Box_{\psi_1}(\text{correctPin})\]

where $\psi_1 = \Box_{\psi_2}(\text{correctPin})$, with $\psi_2 = \Box \Box (\text{cashPay} + \text{returnItems})(\text{correctPin})$
Specification of CoCoME Use Cases 1 and 2
Specific clauses to be specified

**Prop1:** Pay by cash: obligation to swipe the card + correct pin
- Incorrect pin: two more allowed attempts
- After 3 incorrect pins: obligation to pay cash
- No cash: give up the goods
Prop1: Pay by cash: obligation to swipe the card + correct pin
- Incorrect pin: two more allowed attempts
- After 3 incorrect pins: obligation to pay cash
- No cash: give up the goods

Prop2: Normal mode: the cashier may switch to express mode
- If in the last hour 50% of the sales had less than eight items (*)
- In express mode: cashier obliged to eventually go to normal mode
- If (*) holds infinitely often, then the cashier should change to express mode infinitely often
Specification of CoCoME Use Cases 1 and 2
Specific clauses to be specified

**Prop1:** Pay by cash: obligation to swipe the card + correct pin
- Incorrect pin: two more allowed attempts
- After 3 incorrect pins: obligation to pay cash
- No cash: give up the goods

**Prop2:** Normal mode: the cashier may switch to express mode
- If in the last hour 50% of the sales had less than eight items (*)
- In express mode: cashier obliged to eventually go to normal mode
- If (*) holds infinitely often, then the cashier should change to express mode infinitely often

**Prop3:** In express mode: cashier obliged to service customers with less than eight items
- If customer with more than eight items: cashier decides whether to service the client
Specification of Prop1

- Payment with credit card
- Three allowed attempts to enter correct pin
Payment with credit card

Three allowed attempts to enter correct pin

**CSP**: Specification of normal case + refinement to capture exceptional behaviour

- Possible but intricate branching
- Can be described in both **CTL** and **LTL**
- Can be described in **CL** (using CTDs)
Permission to switch from normal to express mode
Obligation to come back to normal mode
Fairness constraint between normal and express mode
Specification of Prop2

- Permission to switch from normal to express mode
- Obligation to come back to normal mode
- Fairness constraint between normal and express mode

Fairness left underspecified in CSP

Cannot be described in CTL (fairness)

Cannot be described in LTL (existential branch)

Can be described in $\mathcal{CL}$ (modulo semantical treatment of fairness)
Specification of Prop3

- Obligation to serve customers with < 8 items
- Permission to service clients with > 8 items
Specification of Prop3

- Obligation to serve customers with < 8 items
- Permission to service clients with > 8 items

Possible in CSP, but process not longer a refinement of original
Described in CTL
Cannot be described in LTL (existential branch)
Can be described in CL
Conclusions

<table>
<thead>
<tr>
<th></th>
<th>LTL</th>
<th>CTL</th>
<th>CSP</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prop1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Prop2</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>(√)</td>
</tr>
<tr>
<td>Prop3</td>
<td>–</td>
<td>✓</td>
<td>(√)</td>
<td>✓</td>
</tr>
</tbody>
</table>

- Different specification languages suitable to different purposes
  - Contracts only for normal behaviour: temporal logic would suffice
  - Composition and comparison of contracts: process calculi more flexibility
- **Operational** approach and **temporal logic** not very suitable to represent certain exceptional cases
  - $CL$ could be encoded in CTL*
- **Deontic** approach suitable to represent obligations, permissions and prohibitions (and many exceptional cases)
  - Needs to be extended to capture more complex compensations (as present in long transactions)