A Formal Language for Electronic Contracts

Cristian Prisacariu — cristi@ifi.uio.no

Gerardo Schneider — gerardo@ifi.uio.no

Precise Modeling and Analysis group (PMA),
University of Oslo

9^{th} IFIP International Conference on
Formal Methods for Open Object-Based Distributed Systems

7^{th} of June 2007, Paphos, Cyprus.

Outline

Aim and Motivation

- Give a formal specification language for specifying contracts.
- The language has a clear and concise syntax
- with formal semantics into a modal logic based on μ-calculus.
- Aims at combining:
  - the logical approach (Deontic Logic) with
  - the automata-like approach
- Tackle the general problem of contracts found in law.
- Language restrictions and design decisions with two aims:
  - capture naturally the clauses found in real-life contracts
  - avoid many of the philosophical paradoxes of Deontic Logic

Ross's paradox:

- "Client is obliged to pay"

  implies
- "Client is obliged to pay or to terminate the contract"

Aim and Motivation

- Give a formal specification language for specifying contracts.
- The language has a clear and concise syntax
- with formal semantics into a modal logic based on μ-calculus.
- Aims at combining:
  - the logical approach (Deontic Logic) with
  - the automata-like approach
- Tackle the general problem of contracts found in law.
- Language restrictions and design decisions with two aims:
  - capture naturally the clauses found in real-life contracts
  - avoid many of the philosophical paradoxes of Deontic Logic

Ross's paradox:

- "Client is obliged to pay"

  implies

- "Client is obliged to pay or to terminate the contract"

Aim and Motivation

why a formal specification language?

Definition

A contract is a document which engages several parties in a transaction and stipulates commitments (obligations, rights, prohibitions), as well as penalties in case of contract violations.

A formal language for contracts should:

- remove the ambiguities of the natural language.
- restrict the user to writing only permitted clauses thus eliminating many of the usual mistakes.
- be able to represent the complex clauses of contracts especially Obligations, Permissions and Prohibitions.
- be amenable to verification by model checking techniques.
- facilitate the (semi-)automatic translation into FSM with the scope of monitoring.

Aim and Motivation

why a formal specification language?

Definition

A contract is a document which engages several parties in a transaction and stipulates commitments (obligations, rights, prohibitions), as well as penalties in case of contract violations.

A formal language for contracts should:

- remove the ambiguities of the natural language.
- restrict the user to writing only permitted clauses thus eliminating many of the usual mistakes.
- be able to represent the complex clauses of contracts especially Obligations, Permissions and Prohibitions.
- be amenable to verification by model checking techniques.
- facilitate the (semi-)automatic translation into FSM with the scope of monitoring.

Deontic Logic

- Deontic logic is the logic of obligation (ought-to), permission, and prohibition.
- Is based on propositional and modal logics.
- *ought-to-do* expressions consider *names of actions*
  "The Internet Provider *ought to send* a password to the Client."
- *ought-to-be* expressions consider *state of affairs* (results of actions)
  "The average bandwidth *ought to be* more than 20kb/s"
- Georg H. von Wright started to sustain a "logic of actions"
  – many of the philosophical paradoxes of Deontic logic are avoided.
- We consider Obligation, Permission and Prohibition only over actions
- We have also assertions which define the "state of affairs"

- Obligation, Permission, and Prohibition are not defined over assertions.

Outline

The Contract Specification Language $\mathcal{CL}$

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

- ϕ denotes assertions and ranges over Boolean expressions including arithmetic comparisons, like “the budget is more than 200\$”.
- $O(\alpha)$, $P(\alpha)$, $F(\delta)$ specify obligation, permission (rights), and prohibition (forbidden) over actions
- α and δ are actions given in the definitions part $\mathcal{D}$.
- $[\alpha]$ and $\langle\alpha\rangle$ are the action parameterized modalities of dynamic logic
- $\mathcal{U}$, $\bigcirc$, and $\Box$ are classical temporal logic operators
- $\wedge$, $\vee$, and $\oplus$ are conjunction, disjunction, and exclusive disjunction

The Contract Specification Language $\mathcal{CL}$

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

- ϕ denotes assertions and ranges over Boolean expressions including arithmetic comparisons, like "the budget is more than 200\$".
- $O(\alpha)$, $P(\alpha)$, $F(\delta)$ specify obligation, permission (rights), and prohibition (forbidden) over actions
- α and δ are actions given in the definitions part $\mathcal{D}$.
- $[\alpha]$ and $\langle\alpha\rangle$ are the action parameterized modalities of dynamic logic
- $\mathcal{U}$, $\bigcirc$, and $\Box$ are classical temporal logic operators
- $\wedge$, $\vee$, and $\oplus$ are conjunction, disjunction, and exclusive disjunction

Actions

- Actions are denoted by α and are constructed using the operators:
  - $+$ choice
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrent execution
- Tests as actions:
  - $\varphi?$ where φ is a contract clause; e.g. an assertion, an obligation, etc.
  - the behavior of a test is like a *guard*; i.e. for action $\varphi? \cdot \alpha$ *if the test succeeds then the action α can be executed*
  - tests are used to model implication:
    $[\varphi?]\mathcal{C}$ is the same as $\varphi \Rightarrow \mathcal{C}$
- Action negation $\overline{\alpha}$
  - with the intuition that 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 $b + a \cdot a$

Actions

- Actions are denoted by α and are constructed using the operators:
  - $+$ choice
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrent execution
- Tests as actions:
  - $\varphi?$ where φ is a contract clause; e.g. an assertion, an obligation, etc.
  - the behavior of a test is like a *guard*; i.e. for action $\varphi? \cdot \alpha$ *if the test succeeds then the action* α *can be executed*
  - tests are used to model implication:
    $[\varphi?]\mathcal{C}$ is the same as $\varphi \Rightarrow \mathcal{C}$
- Action negation $\overline{\alpha}$
  - with the intuition that 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 $b + a \cdot a$

Actions

- Actions are denoted by α and are constructed using the operators:
  - $+$ choice
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrent execution
- Tests as actions:
  - $\varphi?$ where φ is a contract clause; e.g. an assertion, an obligation, etc.
  - the behavior of a test is like a *guard*; i.e. for action $\varphi? \cdot \alpha$ *if the test succeeds then the action* α *can be executed*
  - tests are used to model implication: $[\varphi?]\mathcal{C}$ is the same as $\varphi \Rightarrow \mathcal{C}$
- Action negation $\overline{\alpha}$
  - with the intuition that 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 $b + a \cdot a$

Concurrent actions

- constructed with the & operator: $a\&b$
- "Whenever the Internet traffic is *high* then the client must pay immediately, or the client must notify the service provider by sending an e-mail specifying that he delays the payment."

  $\Box(\phi \implies O(p) \oplus O(d\&n))$
- $O(d\&n) = O(d) \wedge O(n)$
- There may be incompatible actions (which cannot be done at the same time) like: "go west" and "go east". In this case we can have a conflict if we have $O(a) \wedge O(b)$.

Concurrent actions

- constructed with the & operator: $a\&b$
- "Whenever the Internet traffic is *high* then the client must pay immediately, or the client must notify the service provider by sending an e-mail specifying that he delays the payment."

  $\Box(\phi \implies O(p) \oplus O(d\&n))$
- $O(d\&n) = O(d) \wedge O(n)$
- There may be incompatible actions (which cannot be done at the same time) like: "go west" and "go east". In this case we can have a conflict if we have $O(a) \wedge O(b)$.

More on the Contract Language

- Expressing contrary-to-duty (CTDs)
  $O_{\mathcal{C}}(\alpha) = O(\alpha) \wedge [\overline{\alpha}]\mathcal{C}$
- Expressing contrary-to-prohibition (CTPs)
  $F_{\mathcal{C}}(\alpha) = F(\alpha) \wedge [\alpha]\mathcal{C}$
- "In case the client delays the payment, after notification he must immediately lower the Internet traffic to the *low* level, and pay later twice. If the client does not lower the Internet traffic immediately, then the client will have to pay three times."
  $\Box([d\&n](O_{\mathcal{C}}(l) \wedge [l]\Diamond(O(p\&p))$ where $\mathcal{C} = \Diamond O(p\&p\&p)$
- There is a taste of resource-awareness in the actions.
  - Actions like $p\&p$ model discrete values.
  - Even though we have a finite set of atomic actions we get an infinite domain of the compound actions.
  - In work in progress we solve this infiniteness by using so-called *action schemas* (not in this paper)

More on the Contract Language

- Expressing contrary-to-duty (CTDs)
  $O_{\mathcal{C}}(\alpha) = O(\alpha) \wedge [\overline{\alpha}]\mathcal{C}$
- Expressing contrary-to-prohibition (CTPs)
  $F_{\mathcal{C}}(\alpha) = F(\alpha) \wedge [\alpha]\mathcal{C}$
- "In case the client delays the payment, after notification he must immediately lower the Internet traffic to the *low* level, and pay later twice. If the client does not lower the Internet traffic immediately, then the client will have to pay three times."
  $\Box([d\&n](O_{\mathcal{C}}(l) \wedge [l]\Diamond(O(p\&p))$ where $\mathcal{C} = \Diamond O(p\&p\&p)$
- There is a taste of resource-awareness in the actions.
  - Actions like $p\&p$ model discrete values.
  - Even though we have a finite set of atomic actions we get an infinite domain of the compound actions.
  - In work in progress we solve this infiniteness by using so-called *action schemas* (not in this paper)

More on the Contract Language

- Expressing contrary-to-duty (CTDs)
  $O_{\mathcal{C}}(\alpha) = O(\alpha) \wedge [\overline{\alpha}]\mathcal{C}$
- Expressing contrary-to-prohibition (CTPs)
  $F_{\mathcal{C}}(\alpha) = F(\alpha) \wedge [\alpha]\mathcal{C}$
- "In case the client delays the payment, after notification he must immediately lower the Internet traffic to the *low* level, and pay later twice. If the client does not lower the Internet traffic immediately, then the client will have to pay three times."
  $\Box([d\&n](O_{\mathcal{C}}(l) \wedge [l]\Diamond(O(p\&p))$ where $\mathcal{C} = \Diamond O(p\&p\&p)$
- There is a taste of resource-awareness in the actions.
  - Actions like $p\&p$ model discrete values.
  - Even though we have a finite set of atomic actions we get an infinite domain of the compound actions.
  - In work in progress we solve this infiniteness by using so-called *action schemas* (not in this paper)

Outline

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

why μ-calculus?

μ-calculus is a modal logic.

- 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
- 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)
- μ-calculus is a combination of *propositional logic*, the action parameterized *modal operator* $[a]$, and the *fix point constructions*.

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

as the underlying logic

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

Four main differences with respect to the classical μ-calculus:

1. multisets of basic actions as labels: i.e. $\gamma = \{a, a, b\}$ is a label
   $m_\gamma : \mathcal{L} \rightarrow \mathbb{N}$, where $\mathcal{L}$ is the set of basic labels (representing actions)
   e.g.: $m_\gamma(a) = 2$ and $m_\gamma(b) = 1$
2. a set of propositional constants O_a and $\mathcal{F}_a$ one *for each basic action a*
3. a restriction to ensure that there cannot be at the same time an obligation and a prohibition of the same action:
   $||\mathcal{F}_a||^{\mathcal{T}}_{\mathcal{V}} \cap ||O_a||^{\mathcal{T}}_{\mathcal{V}} = \emptyset, \quad \forall a \in \mathcal{L}$
4. a restricted kind of determinism:
   from each state there are no two outgoing arrows labeled with the same action.

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

semantics for the contract language

- semantics for the obligation
  $f^{\mathcal{T}}(O(\&_{i=1}^{n} a_i)) = \langle\{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^{n} O_{a_i})$
  e.g.: $f^{\mathcal{T}}(O(a\&b)) = \langle\{a, b\}\rangle(O_a \wedge O_b)$
  "The Provider is obliged to provide internet and telephony services (at the same time)"
- semantics for the prohibition
  $f^{\mathcal{T}}(F(\&_{i=1}^{n} a_i)) = [\{a_1, \ldots, a_n\}](\wedge_{i=1}^{n} \mathcal{F}_{a_i})$
  e.g.: $f^{\mathcal{T}}(F(a)) = [\{a\}](\mathcal{F}_a)$ often written as just $[a]\mathcal{F}_a$
  "Every action specified in the definition part which is not permitted at one moment is considered forbidden."
- semantics for the permission
  $f^{\mathcal{T}}(P(\&_{i=1}^{n} a_i)) = \langle\{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^{n} \neg\mathcal{F}_{a_i})$
  e.g.: $f^{\mathcal{T}}(P(a)) = \langle a\rangle\neg\mathcal{F}_a$

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

semantics for the contract language

- semantics for the obligation
  $f^{\mathcal{T}}(O(\&_{i=1}^{n} a_i)) = \langle\{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^{n} O_{a_i})$
  e.g.: $f^{\mathcal{T}}(O(a\&b)) = \langle\{a, b\}\rangle(O_a \wedge O_b)$
  "The Provider is obliged to provide internet and telephony services (at the same time)"
- semantics for the prohibition
  $f^{\mathcal{T}}(F(\&_{i=1}^{n} a_i)) = [\{a_1, \ldots, a_n\}](\wedge_{i=1}^{n} \mathcal{F}_{a_i})$
  e.g.: $f^{\mathcal{T}}(F(a)) = [\{a\}](\mathcal{F}_a)$ often written as just $[a]\mathcal{F}_a$
  "Every action specified in the definition part which is not permitted at one moment is considered forbidden."
- semantics for the permission
  $f^{\mathcal{T}}(P(\&_{i=1}^{n} a_i)) = \langle\{a_1, \ldots, a_n\}\rangle(\wedge_{i=1}^{n} \neg\mathcal{F}_{a_i})$
  e.g.: $f^{\mathcal{T}}(P(a)) = \langle a\rangle\neg\mathcal{F}_a$

Outline

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: $O(a) \Rightarrow O(a+b)$

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

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: $O(a) \Rightarrow O(a+b)$

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

Properties of the contract language (II)

- The intuitive implication $O(\alpha) \Rightarrow P(\alpha)$ holds in $\mathcal{CL}$.
  "If the Client is obliged to pay then we can infer that the Client is permitted to do the action of paying."
- The following implications do not hold:
  - $P(a) \not\Rightarrow P(a\&b)$
  - $F(a) \not\Rightarrow F(a\&b)$
    Rights or restrictions on one action do not imply rights or restrictions on executing the action at the same time with another action.
  - $F(a\&b) \not\Rightarrow F(a)$
  - $P(a\&b) \not\Rightarrow P(a)$

Properties of the contract language (II)

- The intuitive implication $O(\alpha) \Rightarrow P(\alpha)$ holds in $\mathcal{CL}$.
  "If the Client is obliged to pay then we can infer that the Client is permitted to do the action of paying."
- The following implications do not hold:
  - $P(a) \nRightarrow P(a\&b)$
  - $F(a) \nRightarrow F(a\&b)$
    Rights or restrictions on one action do not imply rights or restrictions on executing the action at the same time with another action.
  - $F(a\&b) \nRightarrow F(a)$
  - $P(a\&b) \nRightarrow P(a)$

Conclusion

We have seen:

- A formal specification language for contracts with semantics based on a variant of μ-calculus.
- The language
  - is proven to avoid many of the principal deontic paradoxes
  - is specially tailored for specifying contracts
  - combines the logic approach (Deontic Logic) with the automata-like approach.
  - adopts the view of obligations over actions

Further Work

- Model checking of case studies.
- Further theoretical investigations of the underlying actions and the semantics of the contract language.
- Integration into the CREOL object oriented language:
  - Integrate the contract language with the interface specification language of CREOL
  - Use contracts as types to define objects which respect some contract.
  - The formal semantics of CREOL is gine in rewriting logic.
    Good for contract negociation and monitoring.

Thank you!