An Algebraic Structure for Actions found in Contracts

Cristian Prisacariu  
cristi@ifi.uio.no

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

2nd Conference on  
Algebra and Coalgebra in Computer Science,  
Young Researchers Workshop.

20th of August 2007, Bergen, Norway.
Outline

1. Aim and Motivation
2. The contract language $CL$
3. Action Algebra
4. Standard Interpretation as Guarded Rooted Trees
5. Conclusion and Future Work
Aim and Motivation

Our work:

- **Formalizing** and model checking **contracts**
- a formal **Action-Based Contract Language** $\mathcal{CL}$ [FMOODS’07] and
- a model checking attempt [ATVA’07]

In this paper:

- A formal basis for **actions found in contracts**
  - A **complete** action algebra w.r.t. the interpretation
  - Interpretation as guarded rooted trees

- intention to give a direct semantics to $\mathcal{CL}$
Aim and Motivation

Our work:

- **Formalizing** and model checking **contracts**
- a formal **Action-Based Contract Language** $\mathcal{CL}$ [FMOODS’07] and
- a model checking attempt [ATVA’07]

In this paper:

- A formal basis for **actions found in contracts**
  - A **complete** action algebra w.r.t. the interpretation
  - Interpretation as **guarded rooted trees**
- intention to give a **direct semantics** to $\mathcal{CL}$
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.
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.
Outline

1. Aim and Motivation

2. The contract language $\mathcal{CL}$

3. Action Algebra

4. Standard Interpretation as Guarded Rooted Trees

5. Conclusion and Future Work
The Contract Specification Language $\mathcal{CL}$

$\text{Contract} \quad := \quad D \; ; \; C$

$C \quad := \quad \phi \; | \; C_O \; | \; C_P \; | \; C_F \; | \; C \land C \; | \; [\alpha]C \; | \; \langle \alpha \rangle C \; | \; C \cup C \; | \; \Box C \; | \; \Diamond C$

$C_O \quad := \quad O(\alpha) \; | \; C_O \oplus C_O$

$C_P \quad := \quad P(\alpha) \; | \; C_P \oplus C_P$

$C_F \quad := \quad F(\alpha) \; | \; C_F \lor [\alpha]C_F$

- $\phi$ denotes assertions and ranges over Boolean expressions including arithmetic comparisons, like “the budget is more than 200$”.
- $O(\alpha)$, $P(\alpha)$, $F(\alpha)$ specify obligation, permission (rights), and prohibition (forbidden) over actions.
- $\alpha$ are complex actions constructed according to $\mathcal{CA}$ action algebra.
- $[\alpha]$ and $\langle \alpha \rangle$ are the action parameterized modalities of dynamic logic.
- $\cup$, $\Box$, and $\Diamond$ are classical temporal logic operators.
- $\land$, $\lor$, and $\oplus$ are conjunction, disjunction, and exclusive disjunction.
The Contract Specification Language $\mathcal{CL}$

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

- $\phi$ denotes assertions and ranges over Boolean expressions including arithmetic comparisons, like “the budget is more than 200$”.
- $O(\alpha), P(\alpha), F(\alpha)$ specify obligation, permission (rights), and prohibition (forbidden) over actions.
- $\alpha$ are complex actions constructed according to $\mathcal{CA}$ action algebra.
- $[\alpha]$ and $\langle \alpha \rangle$ are the action parameterized modalities of dynamic logic.
- $\mathcal{U}, \bigcirc, \text{ and } \square$ are classical temporal logic operators.
- $\land, \lor, \text{ and } \oplus$ are conjunction, disjunction, and exclusive disjunction.
Outline

1. Aim and Motivation

2. The contract language $\mathcal{CL}$

3. Action Algebra

4. Standard Interpretation as Guarded Rooted Trees

5. Conclusion and Future Work
Actions

- Actions are denoted by $\alpha$ and are constructed using the operators:
  - $+$ choice (idempotent)
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrent execution (not idempotent)
  - basic actions $A_B$ and $0, 1$.

\[
CA = (A, +, \cdot, \&, 0, 1)
\]

- $(A, +, \cdot, 0, 1)$ is an idempotent semiring
- $(A, +, \&, 0, 1)$ is a commutative semiring
- $\&$ shuffles the sequences
  - i.e. an ordered shuffle operator
  - e.g. $(a \cdot b)\&(c \cdot d \cdot e) = a\&c \cdot b\&d \cdot e$
Actions

- **Actions** are denoted by $\alpha$ and are constructed using the operators:
  - $+$ choice (idempotent)
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrent execution (not idempotent)
  - basic actions $A_B$ and $0, 1$.

$$CA = (A, +, \cdot, \&, 0, 1)$$

- $(A, +, \cdot, 0, 1)$ is an idempotent semiring
- $(A, +, \&, 0, 1)$ is a commutative semiring
- $\&$ shuffles the sequences
  - i.e. an ordered shuffle operator
  - e.g. $(a \cdot b)\&(c \cdot d \cdot e) = a\&c \cdot b\&d \cdot e$
Concurrent actions

• constructed with the $\&$ operator: e.g. $d \& n$
• $O(d \& n) = O(d) \land O(n)$
• conflicting actions (cannot be done at the same time) like: “go west” and “go east”; then $O(w) \land O(e)$ is a conflicting clause.
• conflict relation: $a \not\equiv_c b \iff a \& b = 0$
• compatibility relation: $a \sim_c b \iff a \& b \neq 0$, where $a, b \neq 0$

“Whenever the Internet traffic is high ($\phi$) then the client should pay ($p$) double immediately, or the client should notify ($n$) the service provider by sending an e-mail specifying that he delays ($d$) the payment.”

$\Box(\phi \implies O(p \& p) \oplus O(d \& n))$
Concurrent actions

- constructed with the & operator: e.g. $d \& n$
- $O(d \& n) = O(d) \land O(n)$
- conflicting actions (cannot be done at the same time) like: “go west” and “go east”; then $O(w) \land O(e)$ is a conflicting clause.

- conflict relation: $a \#_c b \iff a \& b = 0$
- compatibility relation: $a \sim_c b \iff a \& b \neq 0$, where $a, b \neq 0$

“Whenever the Internet traffic is high ($\phi$) then the client should pay ($p$) double immediately, or the client should notify ($n$) the service provider by sending an e-mail specifying that he delays ($d$) the payment.”

$\Box(\phi \implies O(p \& p) \oplus O(d \& n))$
Concurrent actions

- constructed with the & operator: e.g. \( d \& n \)
- \( O(d \& n) = O(d) \land O(n) \)
- conflicting actions (cannot be done at the same time) like: “go west” and “go east”; then \( O(w) \land O(e) \) is a conflicting clause.

conflict relation : \( a \#_C b \quad \iff \quad a \& b = 0 \)

compatibility relation : \( a \sim_C b \quad \iff \quad a \& b \neq 0 \), where \( a, b \neq 0 \)

“Whenever the Internet traffic is high (\( \phi \)) then the client should pay (\( p \)) double immediately, or the client should notify (\( n \)) the service provider by sending an e-mail specifying that he delays (\( d \)) the payment.”

\[ \square (\phi \implies O(p \& p) \oplus O(d \& n)) \]
Concurrent actions (II)

- & is not idempotent then we have multisets of basic actions
- pomsets of V.Pratt are a generalization of multisets
- 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.
More actions

- **Tests** as actions:
  - \( \varphi? \) where \( \varphi \) 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 \( \alpha \) can be executed
  - algebraically, tests are defined by a Boolean algebra
  - tests are used to model implication:
    \[ [\varphi?]C \] is the same as \( \varphi \Rightarrow C \)

- **Action negation** \( \overline{\alpha} \)
  - with the intuition that it represents all immediate traces that take us outside the trace of \( \alpha \)
  - Involves the use of a *canonic form* of actions
  - E.g.: consider two atomic actions \( a \) and \( b \) then \( a \cdot \overline{a} \) is \( b + a \cdot a \)
More actions

- **Tests** as actions:
  - $\varphi?$ where $\varphi$ 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 $\alpha$ can be executed*
  - algebraically, tests are defined by a Boolean algebra
  - tests are used to model implication:
    
    $[\varphi?]C$ is the same as $\varphi \Rightarrow C$

- **Action negation** $\overline{\alpha}$
  - with the intuition that it represents all immediate traces that take us outside the trace of $\alpha$
  - Involves the use of a *canonic form* of actions
  - E.g.: consider two atomic actions $a$ and $b$ then $\overline{a \cdot b}$ is $b + a \cdot a$
Guarded Rooted Trees

Figure: Examples of finite guarded rooted trees with labeled edges.
Guarded Rooted Trees

Figure: Examples of finite guarded rooted trees with labeled edges.
Guarded Rooted Trees

Figure: Examples of finite guarded rooted trees with labeled edges.
Guarded Rooted Trees

\[ \alpha = \{a, a, b, \ldots\} \]

**Figure:** Examples of finite guarded rooted trees with labeled edges.
Guarded Rooted Trees

\[ \alpha = \{a, a, b, \ldots\} \]

**Figure:** Examples of finite guarded rooted trees with labeled edges.
Interpreting Actions as Guarded Rooted Trees

Figure: Interpreting simple action operators.
Interpreting Actions as Guarded Rooted Trees

1

\[ r \]

\[ \tau \]

\[ n \]

\[ r \]

\[ \alpha \]

\[ n \]

\[ r \]

\[ \alpha \]

\[ \beta \]

\[ n \]

\[ m \]

\[ r \]

\[ \alpha \]

\[ \beta \]

\[ n \]

\[ m \]

\[ \bot \]

\[ r \]

\[ \tau \]

\[ n \]

\[ r \]

\[ \alpha \cdot \beta \]

\[ \alpha + \beta \]

\[ 0 \]

**Figure:** Interpreting simple action operators.
Interpreting Actions as Guarded Rooted Trees

\[ \alpha = \{a\} \]
\[ \beta = \{a, b\} \]

**Figure:** Interpreting simple action operators.
Interpreting Actions as Guarded Rooted Trees

\[
\begin{align*}
&1 & a & a + (a \& b) & a \cdot (a \& b) & 0 \\
&\tau & \{a\} & \{a, b\} & \{a\} & \bot \\
&n & n & m & m & n
\end{align*}
\]

\[\alpha = \{a\} \]
\[\beta = \{a, b\} \]

**Figure:** More concrete actions.
Interpreting Actions as Guarded Rooted Trees

\[ 1 \quad a \& a \quad a + (a \& b) \quad a \cdot (a \& b) \quad 0 \]

\[ \alpha = \{a, a\} \quad \beta = \{a, b\} \]

**Figure:** Concurrent actions as multiset labels.
Tests as Guards

\[ 1 = \top? \]
\[ 0 = \perp? \]

**Figure:** The special tests.
Tests as Guards

Figure: Interpreting simple tests.
Tests as Guards

Figure: Interpreting simple tests.
Tests and Actions

\[ \alpha \cdot \varphi ? \cdot \beta \]

Figure: Sequence of actions and tests.
Tests and Actions

\[ 1 \quad \varphi? \quad \alpha + \varphi? \quad \alpha \cdot \varphi? \cdot \beta \quad 0 \]

\[ \tau \quad \tau \quad \tau \quad \tau \quad \tau \]

\[ r \quad r \quad r \quad r \quad r \]

\[ n \quad n \quad n \quad n \quad n \]

\[ \top \quad \varphi \quad \varphi \quad \varphi \quad \bot \]

\[ \alpha \quad \tau \quad \alpha \quad \tau \quad \varphi \quad \beta \]

\[ m \quad m \quad m \quad m \quad m \]

\[ \alpha \quad \tau \quad \alpha \quad \tau \quad \varphi \quad \beta \]

\[ \tau \quad \tau \quad \tau \quad \tau \quad \tau \]

\[ n \quad n \quad n \quad n \quad n \]

Figure: Nondeterministic choice between actions and tests.
Completeness
w.r.t. the Interpretation

Theorem (Completeness of algebra w.r.t. pruned trees)

The action algebra $CA$ is complete w.r.t. the standard interpretation as pruned guarded rooted trees.

$$\forall \alpha, \beta \in CA$$

$$CA \vdash \alpha = \beta \iff \text{Prune} \circ l_{CA}(\alpha) = \text{Prune} \circ l_{CA}(\beta)$$

- $l_{CA}$: the interpretation function.
- $\text{Prune}$: the pruning function.
Pruning the Trees

- $\alpha \& 1 = \alpha$ in the algebra
- $\alpha \cdot 1 = \alpha$ in the algebra

Figure: Contract $\tau$ labels.
Pruning the Trees

- $\mathbf{\alpha} \& \mathbf{1} = \mathbf{\alpha}$ in the algebra
- $\mathbf{\alpha} \cdot \mathbf{1} = \mathbf{\alpha}$ in the algebra

**Figure:** Remove $\tau$ edges.
Conclusion

We have seen:

- A formal specification language for contracts which has semantics based on a variant of $\mu$-calculus.

- The Action Algebra
  - to help give a direct semantics to $CL$.
  - models actions found in contracts.
  - has a standard interpretation as (pruned) guarded rooted trees which relate to a Kripke-like semantics for $CL$.
  - is complete w.r.t. the interpretation.

- New for contracts:
  - no Kleene start,
  - the action negation,
  - the interpretation as guarded rooted trees with multiset labels to model concurrent actions,
  - additional notions like conflict/compatibility relation, demanding relation, or canonic form.
Conclusion

We have seen:

- A formal specification language for contracts which has semantics based on a variant of $\mu$-calculus.

- The Action Algebra
  - to help give a direct semantics to $\mathcal{L}$.
  - models actions found in contracts.
  - has a standard interpretation as (pruned) guarded rooted trees which relate to a Kripke-like semantics for $\mathcal{L}$.
  - is complete w.r.t. the interpretation.

- New for contracts:
  - no Kleene start,
  - the action negation,
  - the interpretation as guarded rooted trees with multiset labels to model concurrent actions,
  - additional notions like conflict/compatibility relation, demanding relation, or canonic form.
Conclusion

We have seen:

- A formal specification language for contracts which has semantics based on a variant of $\mu$-calculus.

- The Action Algebra
  - to help give a direct semantics to $CL$.
  - models actions found in contracts.
  - has a standard interpretation as (pruned) guarded rooted trees which relate to a Kripke-like semantics for $CL$.
  - is complete w.r.t. the interpretation.

- New for contracts:
  - no Kleene start,
  - the action negation,
  - the interpretation as guarded rooted trees with multiset labels to model concurrent actions,
  - additional notions like conflict/compatibility relation, demanding relation, or canonic form.
Related Work

- D.Kozen et al. on *Kleene algebras* and *dynamic algebras* following J.H.Conway and V.Pratt.
- *mCRL2* of J.F.Groote et al.
- *Dynamic Deontic Logic* of J-J.Meyer.
Further Work

- Model checking of case studies.
- Further theoretical investigations of the semantics of the contract language.
- Enriching the actions with:
  - Durations
  - (simple) types to model subjects
- Enriching the contract language w.r.t. the new actions.
- Extending the contract language with real-time reasoning capabilities as in TCTL (TPTL)
Further Information

- Technical report 361, UiO, on my homepage:
  http://www.ifi.uio.no/~cristi/publications.html
- Introductory course on Kleene Algebras – Dexter Kozen
- Formalizing contracts on Nordunet3 home page:
  http://www.ifi.uio.no/~gerardo/nordunet3/
- These slides on my homepage:
  http://www.ifi.uio.no/~cristi/research.html

Thank you!
References