An Algebraic Structure for Actions found in Contracts

Cristian Prisacariu

cristi@ifi.uio.no

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

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

20^{th} of August 2007, Bergen, Norway.

Outline

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

The Contract Specification Language $\mathcal{CL}$

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

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

The Contract Specification Language $\mathcal{CL}$

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

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

Outline

Actions

- Actions are denoted by α and are constructed using the operators:
  - $+$ choice (idempotent)
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrent execution (not idempotent)
  - basic actions $\mathcal{A}_B$ and $\mathbf{0}$, $\mathbf{1}$.

$$\mathcal{CA} = (\mathcal{A}, +, \cdot, \&, \mathbf{0}, \mathbf{1})$$

  - $(\mathcal{A}, +, \cdot, \mathbf{0}, \mathbf{1})$ is an idempotent semiring
  - $(\mathcal{A}, +, \&, \mathbf{0}, \mathbf{1})$ 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 α and are constructed using the operators:
  - $+$ choice (idempotent)
  - $\cdot$ concatenation (sequencing)
  - $\&$ concurrent execution (not idempotent)
  - basic actions $\mathcal{A}_B$ and $\mathbf{0}$, $\mathbf{1}$.

$$\mathcal{CA} = (\mathcal{A}, +, \cdot, \&, \mathbf{0}, \mathbf{1})$$

  - $(\mathcal{A}, +, \cdot, \mathbf{0}, \mathbf{1})$ is an idempotent semiring
  - $(\mathcal{A}, +, \&, \mathbf{0}, \mathbf{1})$ 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. $d\&n$
- $O(d\&n) = O(d) \wedge O(n)$
- conflicting actions (cannot be done at the same time) like: "*go west*" and "*go east*"; then $O(w) \wedge O(e)$ is a conflicting clause.
- conflict relation : $a \#_{\mathcal{C}} b \stackrel{def}{\Longleftrightarrow} a\&b = \mathbf{0}$
- compatibility relation : $a \sim_{\mathcal{C}} b \stackrel{def}{\Longleftrightarrow} a\&b \neq \mathbf{0}$, where $a, b \neq \mathbf{0}$

- "*Whenever the Internet traffic is high (ϕ) 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. $d\&n$
- $O(d\&n) = O(d) \wedge O(n)$
- conflicting actions (cannot be done at the same time) like: "*go west*" and "*go east*"; then $O(w) \wedge O(e)$ is a conflicting clause.
- conflict relation : $a \#_{\mathcal{C}} b \stackrel{def}{\Longleftrightarrow} a\&b = \mathbf{0}$
- compatibility relation : $a \sim_{\mathcal{C}} b \stackrel{def}{\Longleftrightarrow} a\&b \neq \mathbf{0}$, where $a, b \neq \mathbf{0}$

- "*Whenever the Internet traffic is high* (ϕ) *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. $d\&n$
- $O(d\&n) = O(d) \wedge O(n)$
- conflicting actions (cannot be done at the same time) like: "*go west*" and "*go east*"; then $O(w) \wedge O(e)$ is a conflicting clause.
- conflict relation : $a \#_{\mathcal{C}} b \stackrel{def}{\Longleftrightarrow} a\&b = \mathbf{0}$
- compatibility relation : $a \sim_{\mathcal{C}} b \stackrel{def}{\Longleftrightarrow} a\&b \neq \mathbf{0}$, where $a, b \neq \mathbf{0}$

- "*Whenever the Internet traffic is high (ϕ) 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 (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 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*
  - algebraically, tests are defined by a Boolean algebra
  - 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 $b + a \cdot a$

More actions

- 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*
  - algebraically, tests are defined by a Boolean algebra
  - 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 $b + a \cdot a$

Guarded Rooted Trees

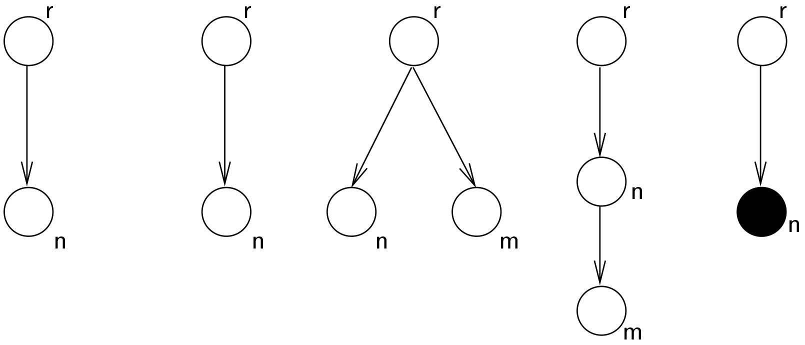


Figure: Examples of finite guarded rooted trees with labeled edges.

Guarded Rooted Trees

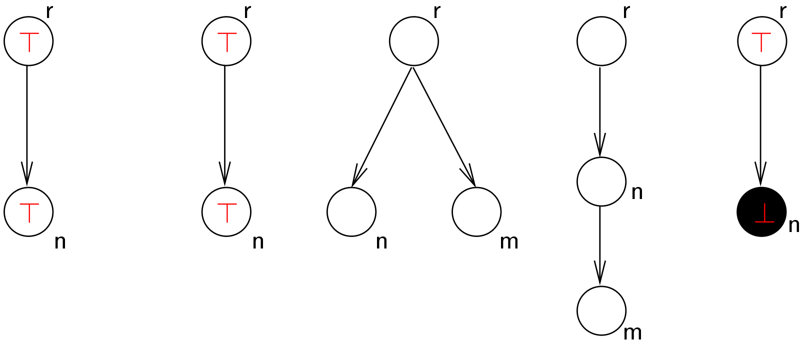


Figure: Examples of finite guarded rooted trees with labeled edges.

Guarded Rooted Trees

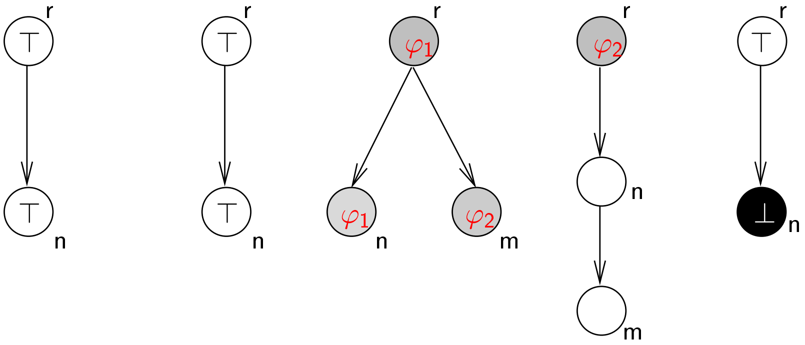


Figure: Examples of finite guarded rooted trees with labeled edges.

Guarded Rooted Trees

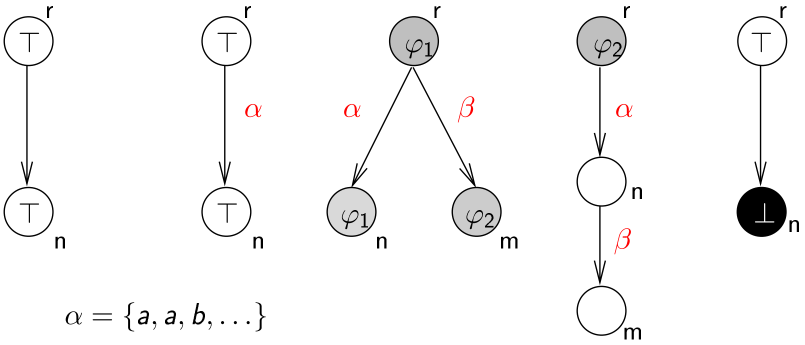
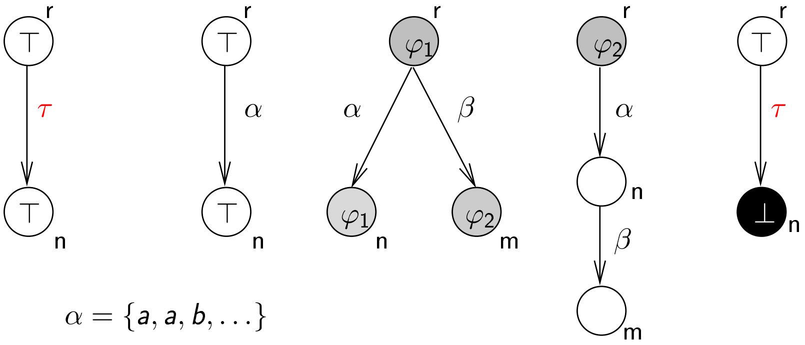


Figure: Examples of finite guarded rooted trees with labeled edges.

Guarded Rooted Trees



$$\alpha = \{a, a, b, \ldots\}$$

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

Interpreting Actions as Guarded Rooted Trees

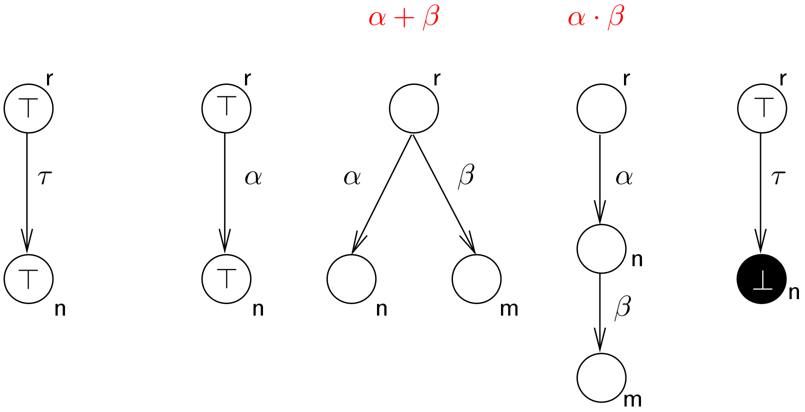


Figure: Interpreting simple action operators.

Interpreting Actions as Guarded Rooted Trees

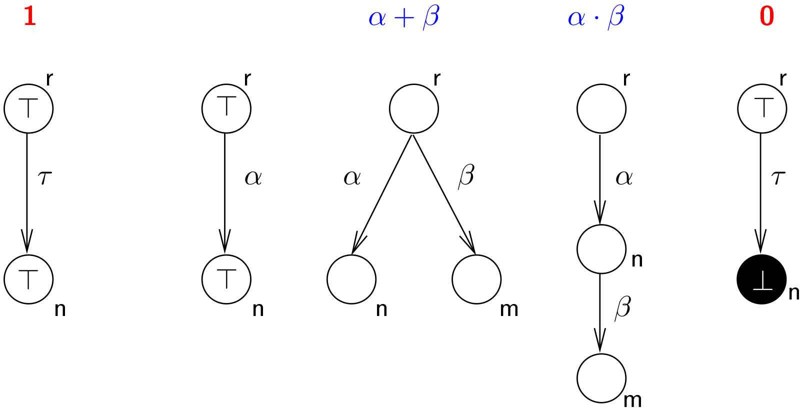
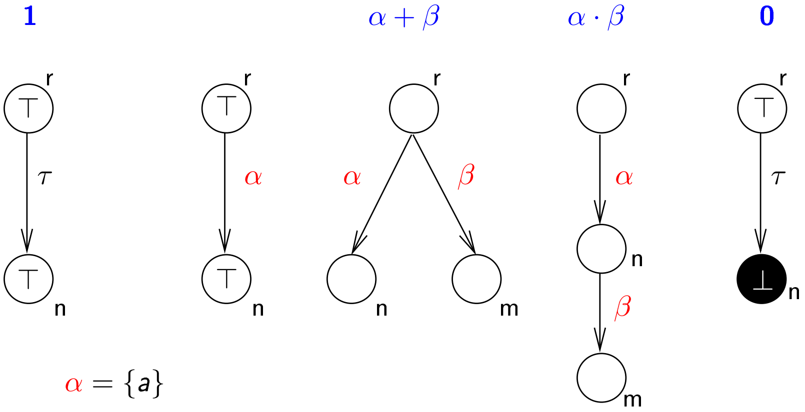


Figure: Interpreting simple action operators.

Interpreting Actions as Guarded Rooted Trees

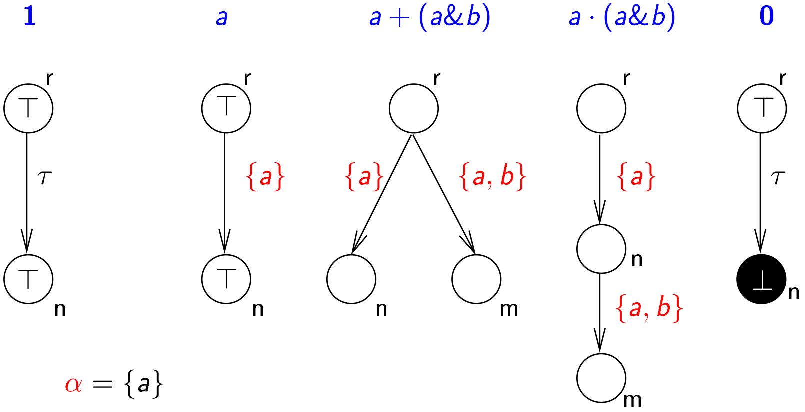


$\alpha = \{a\}$

$\beta = \{a, b\}$

Figure: Interpreting simple action operators.

Interpreting Actions as Guarded Rooted Trees

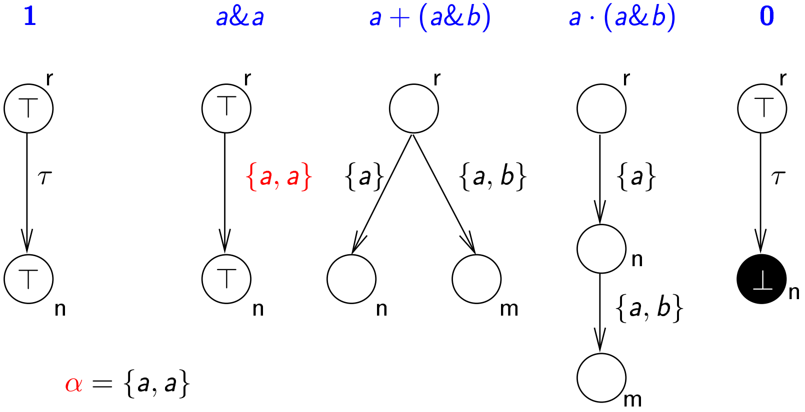


$\alpha = \{a\}$

$\beta = \{a, b\}$

Figure: More concrete actions.

Interpreting Actions as Guarded Rooted Trees



$\alpha = \{a, a\}$

$\beta = \{a, b\}$

Figure: Concurrent actions as multiset labels.

Tests as Guards

$\mathbf{1} = \top?$

$\mathbf{0} = \bot?$

Figure: The special tests.

Tests as Guards

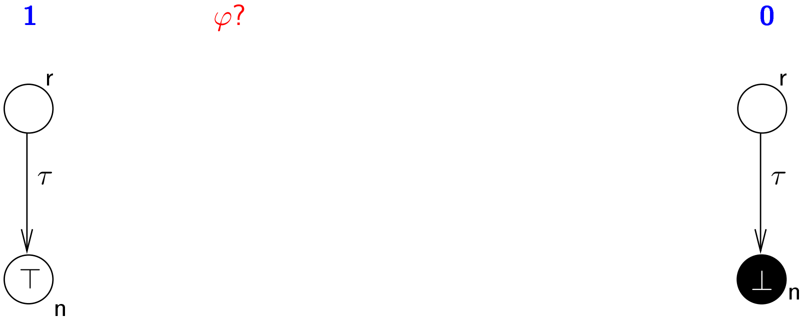


Figure: Interpreting simple tests.

Tests as Guards

Figure: Interpreting simple tests.

Tests and Actions

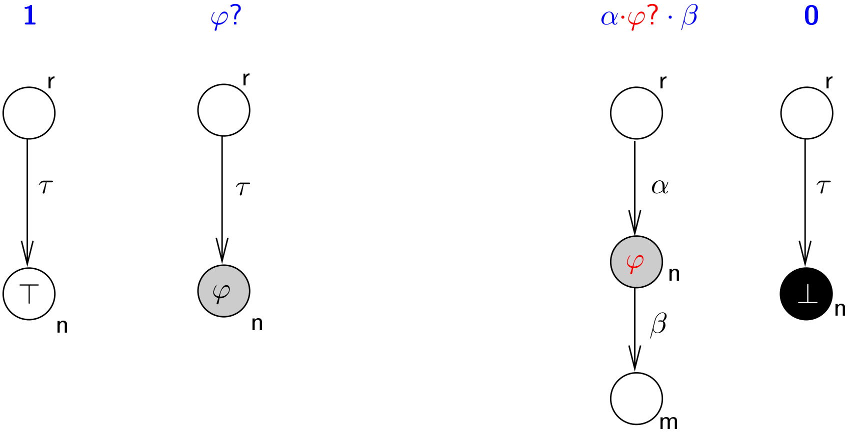


Figure: Sequence of actions and tests.

Tests and Actions

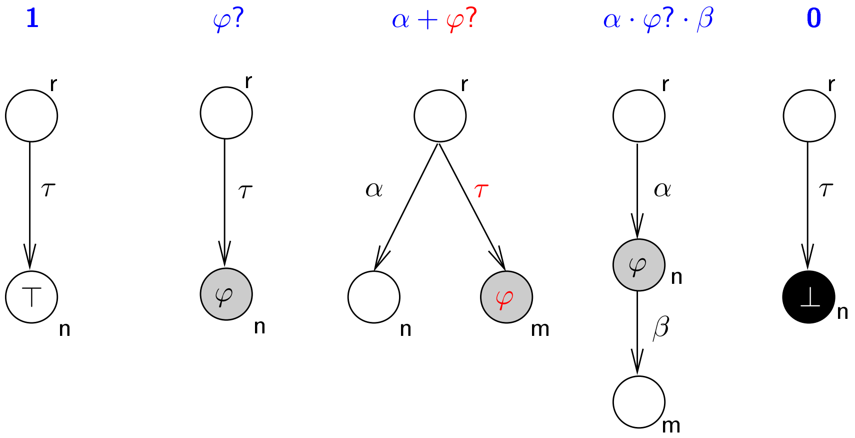


Figure: Nondeterministic choice between actions and tests.

Completeness

w.r.t. the Interpretation

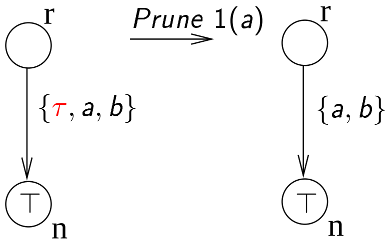
Theorem (Completeness of algebra w.r.t. pruned trees)

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

$$\begin{array}{ccc} & & \forall\alpha,\beta\in\mathcal{CA} \\ \mathcal{CA}\vdash\alpha=\beta & \Leftrightarrow & Prune\circ I_{\mathcal{CA}}(\alpha)=Prune\circ I_{\mathcal{CA}}(\beta) \end{array}$$

- $I_{\mathcal{CA}}$: the interpretation function.
- *Prune* : the pruning function.

Pruning the Trees



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

Figure: Contract τ labels.

Pruning the Trees

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

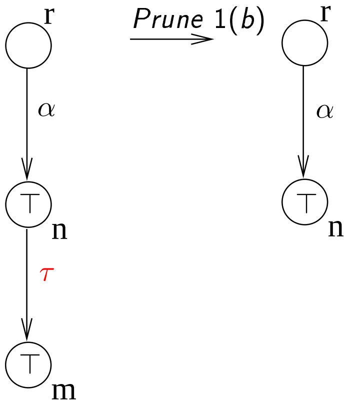


Figure: Remove τ edges.

Conclusion

We have seen:

- A formal specification language for contracts
  which has semantics based on a variant of μ-calculus .
- The Action Algebra
  - to help give a direct semantics to $\mathcal{CL}$.
  - models actions found in contracts.
  - has a standard interpretation as (*pruned*) guarded rooted trees which relate to a Kripke-like semantics for $\mathcal{CL}$.
  - 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 μ-calculus .
- The Action Algebra
  - to help give a direct semantics to $\mathcal{CL}$.
  - models actions found in contracts.
  - has a standard interpretation as (*pruned*) guarded rooted trees which relate to a Kripke-like semantics for $\mathcal{CL}$.
  - 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 μ-calculus .
- The Action Algebra
  - to help give a direct semantics to $\mathcal{CL}$.
  - models actions found in contracts.
  - has a standard interpretation as (*pruned*) guarded rooted trees which relate to a Kripke-like semantics for $\mathcal{CL}$.
  - 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