

New Proof Obligations

Benjamin Weiß

Institut für Theoretische Informatik
Universität Karlsruhe

06.06.2006

- Result of OCL (or JML) translation:

- Result of OCL (or JML) translation:
 - Class invariants as FOL formulas

- Result of OCL (or JML) translation:
 - Class invariants as FOL formulas
 - Operation contracts, consisting of:
 - Precondition as FOL formula
 - Postcondition as FOL formula
 - Modifier set
 - Termination marker *partial* or *total*

- Result of OCL (or JML) translation:
 - Class invariants as FOL formulas
 - Operation contracts, consisting of:
 - Precondition as FOL formula
 - Postcondition as FOL formula
 - Modifier set
 - Termination marker *partial* or *total*
 - Classes

- Result of OCL (or JML) translation:
 - Class invariants as FOL formulas
 - Operation contracts, consisting of:
 - Precondition as FOL formula
 - Postcondition as FOL formula
 - Modifier set
 - Termination marker *partial* or *total*
 - Classes
 - Operations

- Result of OCL (or JML) translation:
 - Class invariants as FOL formulas
 - Operation contracts, consisting of:
 - Precondition as FOL formula
 - Postcondition as FOL formula
 - Modifier set
 - Termination marker *partial* or *total*
 - Classes
 - Operations
- Goal: Proof obligations, i.e. sets of DL formulas

- Result of OCL (or JML) translation:
 - Class invariants as FOL formulas
 - Operation contracts, consisting of:
 - Precondition as FOL formula
 - Postcondition as FOL formula
 - Modifier set
 - Termination marker *partial* or *total*
 - Classes
 - Operations
- Goal: Proof obligations, i.e. sets of DL formulas
- Proof obligation template: A template instantiating to a proof obligation, e.g. *EnsuresPost(opct, AssumedInvs)*

- Result of OCL (or JML) translation:
 - Class invariants as FOL formulas
 - Operation contracts, consisting of:
 - Precondition as FOL formula
 - Postcondition as FOL formula
 - Modifier set
 - Termination marker *partial* or *total*
 - Classes
 - Operations
- Goal: Proof obligations, i.e. sets of DL formulas
- Proof obligation template: A template instantiating to a proof obligation, e.g. *EnsuresPost(opct, AssumedInvs)*
- Revised templates defined in KeY book chapter “Proof Obligations” by Andreas Roth

- *DisjointPreconditions*($opct_1, opct_2$)

Horizontal Proof Obligation Templates

- *DisjointPreconditions*($opct_1, opct_2$)
- *BehaviouralSubtypingInv*(C, D)

Horizontal Proof Obligation Templates

- *DisjointPreconditions*($opct_1, opct_2$)
- *BehaviouralSubtypingInv*(C, D)
- *BehaviouralSubtypingOpPair*($opct_C, opct_D$)

Horizontal Proof Obligation Templates

- *DisjointPreconditions*($opct_1, opct_2$)
- *BehaviouralSubtypingInv*(C, D)
- *BehaviouralSubtypingOpPair*($opct_C, opct_D$)
- *BehaviouralSubtypingOp*(C, D)

Horizontal Proof Obligation Templates

- *DisjointPreconditions*($opct_1, opct_2$)
- *BehaviouralSubtypingInv*(C, D)
- *BehaviouralSubtypingOpPair*($opct_C, opct_D$)
- *BehaviouralSubtypingOp*(C, D)
- *StrongOperationContract*($opct, AssumedInvs, EnsuredInvs$)

Vertical Proof Obligations: General Structure

$ValidCall_{op} \wedge Assumptions \rightarrow [Prg_{op}]Assertions$

Vertical Proof Obligations: General Structure

$ValidCall_{op} \wedge Assumptions \rightarrow [Prg_{op}]Assertions$

$ValidCall_{op} = LHS \wedge self.<created> \doteq TRUE \wedge self \neq null$
 $\wedge \bigwedge_i (p_i.<created> \doteq TRUE \vee p_i \doteq null)$

Vertical Proof Obligations: General Structure

$ValidCall_{op} \wedge Assumptions \rightarrow [Prg_{op}] Assertions$

$ValidCall_{op} = LHS \wedge self.<created> \doteq TRUE \wedge self \neq null$
 $\wedge \bigwedge_i (p_i.<created> \doteq TRUE \vee p_i \doteq null)$

Vertical Proof Obligations: General Structure

$ValidCall_{op} \wedge Assumptions \rightarrow [Prg_{op}]Assertions$

$ValidCall_{op} = LHS \wedge self.<created> \doteq TRUE \wedge self \neq null$
 $\wedge \bigwedge_i (p_i.<created> \doteq TRUE \vee p_i \doteq null)$

$Prg_{op} = \left\{ \begin{array}{l} T_1 p'_1 = p_1; \\ \vdots \\ T_n p'_n = p_n; \\ exc = null; \\ \text{try } \{ \\ \quad \text{result} = self.op(p'_1, \dots, p'_n) @ C; \\ \} \text{ catch(Throwable } e) \{ \\ \quad exc = e; \\ \} \end{array} \right.$

- *EnsuresPost*(*opct*, *AssumedInvs*)

Vertical Proof Obligation Templates

- *EnsuresPost*(*opct*, *AssumedInvs*)
- *InitInv*(*Invs*)

Vertical Proof Obligation Templates

- *EnsuresPost*(*opct*, *AssumedInvs*)
- *InitInv*(*Invs*)
- *PreservesInv*(*op*, *AssumedInvs*, *EnsuredInvs*)

Vertical Proof Obligation Templates

- $EnsuresPost(opct, AssumedInvs)$
- $InitInv(Invs)$
- $PreservesInv(op, AssumedInvs, EnsuredInvs)$
- $PreservesOwnInv(op, AssumedInvs)$

Vertical Proof Obligation Templates

- *EnsuresPost*(*opct*, *AssumedInvs*)
- *InitInv*(*Invs*)
- *PreservesInv*(*op*, *AssumedInvs*, *EnsuredInvs*)
- *PreservesOwnInv*(*op*, *AssumedInvs*)
- *PreservesGuard*(*op*, *Invs*, *GuardClasses*)

Vertical Proof Obligation Templates

- *EnsuresPost*(*opct*, *AssumedInvs*)
- *InitInv*(*Invs*)
- *PreservesInv*(*op*, *AssumedInvs*, *EnsuredInvs*)
- *PreservesOwnInv*(*op*, *AssumedInvs*)
- *PreservesGuard*(*op*, *Invs*, *GuardClasses*)
- *RespectsModifies*(*opct*, *AssumedInvs*)

RespectsModifies: Example

Modifier set $\{a\}$:

$$\begin{aligned} & \text{ValidCall}_{op} \wedge \text{AssumedInvs} \wedge \text{precondition} \wedge \\ & \{a := a'\} \langle \text{anon}() ; \rangle \text{true} \\ & \rightarrow \\ & [\text{Prg}_{op}] \{a := a'\} \langle \text{anon}() ; \rangle \text{true} \end{aligned}$$

RespectsModifies: Example

Modifier set $\{a\}$:

$$\begin{aligned} & \textit{ValidCall}_{op} \wedge \textit{AssumedInvs} \wedge \textit{precondition} \wedge \\ & \{a := a'\} \langle \textit{anon}() ; \rangle \textit{true} \\ & \rightarrow \\ & [\textit{Prg}_{op}] \{a := a'\} \langle \textit{anon}() ; \rangle \textit{true} \end{aligned}$$

RespectsModifies: Example

Modifier set $\{a\}$:

$$\begin{aligned} & \text{ValidCall}_{op} \wedge \text{AssumedInvs} \wedge \text{precondition} \wedge \\ & \{a := a'\} \langle \text{anon}() ; \rangle \text{true} \\ & \rightarrow \\ & [\text{Prg}_{op}] \{a := a'\} \langle \text{anon}() ; \rangle \text{true} \end{aligned}$$

RespectsModifies: Example

Modifier set $\{a\}$:

$$\begin{aligned} & \text{ValidCall}_{op} \wedge \text{AssumedInvs} \wedge \text{precondition} \wedge \\ & \{a := a'\} \langle \text{anon}() ; \rangle \text{true} \\ & \rightarrow \\ & [\text{Prg}_{op}] \{a := a'\} \langle \text{anon}() ; \rangle \text{true} \end{aligned}$$

RespectsModifies: Another Example

Modifier set $\{a, f(a)\}$:

$$\text{ValidCall}_{op} \wedge \text{AssumedInvs} \wedge \text{precondition} \wedge a^{\text{@pre}} = a \wedge$$
$$\{f(a) := f'(a), a := a'\} \langle \text{anon}(); \rangle \text{true}$$

\rightarrow

$$[\text{Prg}_{op}] \{f(a^{\text{@pre}}) := f'(a^{\text{@pre}}), a := a'\} \langle \text{anon}(); \rangle \text{true}$$

RespectsModifies: Another Example

Modifier set $\{a, f(a)\}$:

$$\text{ValidCall}_{op} \wedge \text{AssumedInvs} \wedge \text{precondition} \wedge a^{\text{pre}} = a \wedge \\ \{f(a) := f'(a), a := a'\} \langle \text{anon}(); \rangle \text{true}$$

→

$$[\text{Prg}_{op}] \{f(a^{\text{pre}}) := f'(a^{\text{pre}}), a := a'\} \langle \text{anon}(); \rangle \text{true}$$

RespectsModifies: Another Example

Modifier set $\{a, f(a)\}$:

$$\text{ValidCall}_{op} \wedge \text{AssumedInvs} \wedge \text{precondition} \wedge a^{\text{@pre}} = a \wedge \\ \{f(a) := f'(a), a := a'\} \langle \text{anon}(); \rangle \text{true}$$

→

$$[\text{Prg}_{op}] \{f(a^{\text{@pre}}) := f'(a^{\text{@pre}}), a := a'\} \langle \text{anon}(); \rangle \text{true}$$

- Horizontal proof obligation templates:
 - *DisjointPreconditions*($opct_1, opct_2$)
 - *BehaviouralSubtypingInv*(C, D)
 - *BehaviouralSubtypingOpPair*($opct_C, opct_D$)
 - *BehaviouralSubtypingOp*(C, D)
 - *StrongOperationContract*($opct, AssumedInvs, EnsuredInvs$)
- Vertical proof obligation templates:
 - *EnsuresPost*($opct, AssumedInvs$)
 - *InitInv*($Invs$)
 - *PreservesInv*($op, AssumedInvs, EnsuredInvs$)
 - *PreservesOwnInv*($op, AssumedInvs$)
 - *PreservesGuard*($op, Invs, GuardClasses$)
 - *RespectsModifies*($opct, AssumedInvs$)