Contents

- Overview of KeY
- UML and its semantics
- Introduction to OCL
- Specifying requirements with OCL
- Modelling of Systems with Formal Semantics
- Propositional & First-order logic, sequent calculus
- OCL to Logic, horizontal proof obligations, using KeY
- Dynamic logic, proving program correctness
- Java Card DL
- Vertical proof obligations, using KeY
- Wrap-up, trends
Formal Verification

Real World

UML
OCL
Java

Obj. Diagr.
Snapshot/
LTS

FO Logic
Dyn. Logic

FO Model
Kripke Str.

Sequent
Calculus

Translation

I, ⊨
Classifier Context (Invariants)

context typeName

inv 'Boolean OclExpression-with-self'
OCL Context Declarations as Universal Quantifiers

Classifier Context (Invariants)

context typeName
    inv 'Boolean OclExpression-with-self'

Equivalent to universally quantified expression

inv typeName.allInstances() -> forall(x | OclExpression-with-x)
Classifier Context (Invariants)

context typeName

inv ‘Boolean OclExpression-with-self’

Equivalent to universally quantified expression

inv typeName.allInstances() \rightarrow \forall x \mid OclExpression-with-x

Example

context Person

inv self.age >= 0 \Rightarrow inv Person.allInstances() \rightarrow \forall x \mid x.age >= 0
Universally quantified OCL expression

\[ \text{inv } \text{typeName.allInstances} \rightarrow \text{forall}(x \mid \text{OclExpression-with-x}) \]
Universally quantified OCL expression

\[ \text{inv } \text{typeName}.allInstances \rightarrow \text{forall}(x \mid \text{OclExpression-with-x}) \]

Translation \( T \) to universal quantifier over variable \( x \) of type \text{typeName}

\[ \forall x.T(\text{OclExpression-with-x}) \]
Translating Universal Quantifiers from OCL to FOL

Universally quantified OCL expression

\[ \text{inv} \quad \text{typeName}.allInstances \rightarrow \forall x \mid \text{OclExpression-with-}x \]

Translation \( T \) to universal quantifier over variable \( x \) of type \( \text{typeName} \)

\[ \forall x. T(\text{OclExpression-with-}x) \]

Example

\[ \text{inv} \quad \text{Person.allInstances()} \rightarrow \forall x \mid x.\text{age} \geq 0 \]

\[ \Rightarrow \quad x : \text{Person} \quad \forall x.(T(x.\text{age} \geq 0)) \]
Quantification over Existing Objects

If $x$ is variable of type $C$ from UML context,
then $\forall x.\phi$ quantifies over all objects typeable with $C$

We want only the created objects in the current snapshot!
Quantification over Existing Objects

If $x$ is variable of type $C$ from UML context, then $\forall x.\phi$ quantifies over all objects typeable with $C$

We want only the created objects in the current snapshot!

Assume that each class $C$ has Boolean attribute $<\text{created}>$

$I(<\text{created}>)(o)$ is true iff $o$ has been created in state described by $I$
Quantification over Existing Objects

If $x$ is variable of type $C$ from UML context, then $\forall x.\phi$ quantifies over all objects typeable with $C$

We want only the created objects in the current snapshot!

Assume that each class $C$ has Boolean attribute $<\text{created}>$

$I(<\text{created}>)(o)$ is true iff $o$ has been created in state described by $I$

Instead of $\forall$ use quantifier $\dot{\forall}$ defined as:

$$\dot{\forall} x.\phi \iff \forall x.(x.<\text{created}> \rightarrow \phi)$$

Instead of $\exists$ use quantifier $\dot{\exists}$ defined as:

$$\dot{\exists} x.\phi \iff \exists x.(x.<\text{created}> \& \phi)$$
Translating OCL to FOL: Attributes

Attributes

OCL constraint with attribute

\[ x.age \geq 0 \]
Translating OCL to FOL: Attributes

Attributes

OCL constraint with attribute

\[ x.\text{age} \geq 0 \]

UML attribute semantics

\[ I(\text{age}) \text{ function from } I(\text{Person}) \text{ to } I(\text{int}) \]
Translating OCL to FOL: Attributes

Attributes

OCL constraint with attribute

\[ x.\text{age} \geq 0 \]

UML attribute semantics

\[ I(\text{age}) \text{ function from } I(\text{Person}) \text{ to } I(\text{int}) \]

FOL type hierarchy & signature (fragment)

\[ T = \{ \text{Person}, \ldots, \text{int}, \ldots \} \]

\[ \text{FSym} = \{ \text{age} \} \text{ with } \text{age} : \text{Person} \rightarrow \text{int} \]

\[ \text{PSym} = \{ \geq, \leq, >, <, \ldots \} \]
Translating OCL to FOL: Attributes

**Attributes**

OCL constraint with attribute

\[ x.\text{age} \geq 0 \]

**UML attribute semantics**

\( I(\text{age}) \) function from \( I(\text{Person}) \) to \( I(\text{int}) \)

**FOL type hierarchy & signature (fragment)**

\[ T = \{ \text{Person}, \ldots, \text{int}, \ldots \} \]

\[ \text{FSym} = \{ \text{age} \} \text{ with } \text{age} : \text{Person} \rightarrow \text{int} \]

\[ \text{PSym} = \{ \geq, \leq, >, <, \ldots \} \]

**FOL translation**

\[ T(x.\text{age} \geq 0) = \text{age}(x) \geq 0 \]
Notational Conventions

Allow postfix-dot notation for functions that model attributes

Example

\[ \text{age}(x) \geq 0 \quad \Rightarrow \quad x.\text{age} \geq 0 \]

In simple cases FOL translation looks exactly like OCL:

OCL expressions w/o iterators are alternative concrete syntax of FOL
Notational Conventions

Allow postfix-dot notation for functions that model attributes

Example

\[ \text{age}(x) \geq 0 \quad \Rightarrow \quad x.\text{age} \geq 0 \]

In simple cases FOL translation looks exactly like OCL:
OCL expressions w/o iterators are alternative concrete syntax of FOL

No generic types in Java Card and FOL (such as \( \text{Set}(\text{Person}) \))
Translation generates suitable flat types on-the-fly
SetOfPerson, SequenceOfPerson, etc.

Shorthand for sets of objects: Vehicle\{\}, Person\{\}, int\{\}
Assorted Remarks

- FOL translation of OCL attribute interpreted as total function
  Value of an attribute might be **null**

- Symbols with fixed interpretation for many OCL properties
  $\leq$, size, includes, $+$, 17, self, result, **etc.**

Correct intended semantics guaranteed by sound calculus rules (automatically loaded)

- If owner type of functions that model attributes and operations is required to resolve overloading, then write it **in front**:

  Person ::= age(x), Person{} ::= includes(siblings(self), p)
Translating OCL to FOL: Associations

Associations

Multiplicity 1: like attributes, but no dot notation

Function \( \langle \text{supplier-role-name} \rangle : \langle \text{client-type} \rangle \rightarrow \langle \text{supplier-type} \rangle \)

Example: father : Person \( \rightarrow \) Person

Use explicit role name if present, otherwise default role name

\[
\text{not}(\text{self.father} = \text{self.mother}) \quad \overset{T}{\Rightarrow} \quad \neg(\text{father}(self) = \text{mother}(self))
\]
Associations

Other multiplicity than 1:

**Function** \( \langle \text{supplier-role-name} \rangle : \langle \text{client-type} \rangle \rightarrow \langle \text{Supplier-type}\{\} \rangle \)
Translating OCL to FOL: Associations

Associations

Other multiplicity than 1:

Function \( \langle \text{supplier-role-name} \rangle : \langle \text{client-type} \rangle \rightarrow \langle \text{Supplier-type}\{\} \rangle \)

Example: siblings : Person \(\rightarrow\) Person\{\}

self.siblings = self.nephews \(\Rightarrow\) siblings\(\(self) \uprightighteq\) nephews\(\(self)\)

Problem: no rules for equality of sets of objects \(\Rightarrow\) extensionality
Translating OCL to FOL: Associations

Associations

Other multiplicity than 1:

Function \( \langle \text{supplier-role-name} \rangle : \langle \text{client-type} \rangle \rightarrow \langle \text{Supplier-type} \{\} \rangle \)

\( \text{siblings}(self) \models \text{nephews}(self) \) expanded into:

\[ \forall p. (\text{Person}\{\}::\text{includes}(\text{siblings}(self), p)) \leftrightarrow (\text{Person}\{\}::\text{includes}(\text{nephews}(self), p)) \]
Translating OCL to FOL: allInstances()

**allInstances()**

**Argument of OCL quantifier** `forall`, `exists`  
Analogous treatment to class context declaration

**Example**  
Person.allInstances() -> `forall(age >= 0)`  
\[ \forall x. (x.age \geq 0) \]
Translating OCL to FOL: allInstances()

allInstances()

Other collection property than quantifier

For \( T\.allInstances() \) create constant \( T\{\}::allInstances : \rightarrow T\{\} \)

Add “definition” of \( T\{\}::allInstances \) to goal antecedent:

\[
\forall x. \ T\{\}::includes(T\{\}::allInstances, x)
\]

Example for translation of allInstances()

\[
T \Rightarrow \text{Person.allInstances()} \rightarrow \text{size()} = 1
\]

\[
\text{Person}\{}::size(\text{Person}\{}::allInstances) \Rightarrow 1
\]
Translating OCL to FOL: Important Issues

- In many cases FOL translation follows OCL closely
- Some collection properties have complicated translations (select, reject)
  Translator optimizes whenever possible
- Sometimes, translation declares new function symbols
  Definitions placed in antecedent (ie, left) of sequent arrow ==> 
- Details of translation (see also course web page):
  B. Beckert, U. Keller, P Schmitt: Translating the OCL into First-order Predicate Logic
  A. Roth & P. Schmitt
  Formal Specification, Section 5.2.3
Horizontal Verification: Behavioural Subtyping

**Substitution principle** (Liskov, 1993)

Let $\phi$ be a property provable about objects $x$ of type $T$. Then $\phi$ should be true for objects $y$ of type $S$ where $S \subseteq T$. 
Substitution principle (Liskov, 1993)

Let $\phi$ be a property provable about objects $x$ of type $T$. Then $\phi$ should be true for objects $y$ of type $S$ where $S \subseteq T$.

Consequence is invariant subtyping property:

Invariant of a class must imply invariant of all parent classes.
**Substitution principle** (Liskov, 1993)

Let $\phi$ be a property provable about objects $x$ of type $T$. Then $\phi$ should be true for objects $y$ of type $S$ where $S \subseteq T$.

Consequence is **invariant subtyping** property:

Invariant of a class must imply invariant of all parent classes

$\text{inv}_S$ is (FOL translation of) OCL invariant constraint of a class $S$

$T_1, \ldots, T_n$ parent classes and interfaces of $S$

**Proof obligation:** $\forall self. (\text{inv}_S \rightarrow (\text{inv}_{T_1} \& \cdots \& \text{inv}_{T_n}))$