USING THE BASE SEMANTICS GIVEN BY FUML FOR VERIFICATION

Presenter: Yu Bai
bai@cs.uni-kl.de

Alessandro Gerlinger Romero
romgerale@yahoo.com.br

Klaus Schneider

Maurício Gonçalves Vieira Ferreira
Brazilian National Institute for Space Research – INPE – Brazil

Department of Computer Science – University of Kaiserslautern – Germany

International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2014
Lisbon, Portugal, January 7-9, 2014
Using the Base Semantics given by fUML for Verification

Agenda

• Introduction
  ▪ Motivation
  ▪ Goal

• Context

• Verification
  ▪ Example
  ▪ Syntax
  ▪ Semantics
  ▪ Theorem proving

• Conclusions
Using the Base Semantics given by fUML for Verification

Agenda

• Introduction
  ▪ Motivation
  ▪ Goal
• Context
• Verification
  ▪ Example
  ▪ Syntax
  ▪ Semantics
  ▪ Theorem proving
• Conclusions
Motivation

• UML supports a large number of Model-Driven Development (MDD) methods

  • UML is expressive, but the lack of formal foundations in the UML results in imprecise models

• A major focus in systems and software engineering has been how to introduce precision in the approaches based on UML through formal methods.

  • There is a large number of research papers about semantics for models defined using UML (mainly, operational semantics and translation)

  • Object Management group has the effort Foundational Subset for Executable UML Models (fUML)
Goal

• **Question:** Does fUML allow the use of model-checking or theorem proving?

• **Goal:** evaluate a subset of base semantics (part of fUML) covering from the formal definition up to how to use it for theorem proving.

The initial results show that the base semantics, when mature, can play an important role in the formal verification of UML models.
Using the Base Semantics given by fUML for Verification

Agenda

• Introduction
  ▪ Motivation
  ▪ Goal

• Context

• Verification
  ▪ Example
  ▪ Syntax
  ▪ Semantics
  ▪ Theorem proving

• Conclusions
Using the Base Semantics given by fUML for Verification

OMG Specifications – fUML and Alf

- **fUML** is a subset of UML consisting of the key parts of UML **activities** and classes.
- **Alf** is the **textual concrete syntax** for the abstract action language defined by fUML.
OMG Specifications – Base semantics given by fUML (1/2)

- Language = (AbstractSyntax, ConcreteSyntax, SemanticDomain, SyntacticMapping, SemanticMapping)
OMG Specifications – Base semantics given by fUML

- fUML defines **two** semantics, the **formal** one (base semantics) is encoded using CLIF (Common Logic Interchange Format).

\[
\forall (n1 \ n2) \ (\text{iff} \ (\text{form:same-syntactic-container} \ n1 \ n2) \\
\quad \ (\exists c) \ (\text{or} \\
\quad \quad \quad \ (\text{and} \ (\text{buml:inStructuredNode} \ n1 \ c) \\
\quad \quad \quad \quad \ (\text{buml:inStructuredNode} \ n2 \ c)) \\
\quad \quad \quad \ (\text{and} \ (\text{buml:activity} \ n1 \ c) \\
\quad \quad \quad \quad \ (\text{buml:activity} \ n2 \ c))))
\]

- What is the purpose of the **base semantics**?

  - OMG answer
    - The conformance for an interpreter would be demonstrated by a **formal proof** that it respects all the definitions of the base semantics.

  - Another possible answer
    - It can be used to evaluate **formally properties of UML activities**.
Using the Base Semantics given by fUML for Verification

Agenda

• Introduction
  ▪ Motivation
  ▪ Goal
• Context
• Verification
  ▪ Example
  ▪ Syntax
  ▪ Semantics
  ▪ Theorem proving
• Conclusions
Using the Base Semantics given by fUML for Verification

Verification - Example

User Definition using Abstract Syntax

(1) activity Main: diagram Main

ValueSpecificationAction1

variableX

ValueSpecificationAction1

Type LiteralInteger
Name: LiteralInteger1
Value: 1

true

ObjectFlow1

Main: Activity

ObjectFlow1: ObjectFlow

ValueSpecificationAction1: ValueSpecificationAction

Variable_x_ForkNode: ForkNode

LiteralInteger1: LiteralInteger

variableX: OutputPin

value = 1

Semantic Domain for the User Definition

(II) - BS

(buml:ValueSpecificationAction Main:ValueSpecificationAction1)
(buml:activity Main:ValueSpecificationAction1 Main)
(buml:value Main:ValueSpecificationAction1:ValueSpecification.LiteralInteger1 form:1...

(III) - BS

subjectOcc1: ValueSpecificationActionActivation

node = ValueSpecificationAction1

value = IntegerValue

value = 1

(V) - BS

actOcc1: Main

Main:ValueSpecification1: subjectOcc1

Main:ValueSpecificationAction1:variableX: 1

(form:property-value actOcc1 Main:ValueSpecificationAction1 subjectOcc1 state)
(form:property-value actOcc1 Main:ValueSpecificationAction1:variableX form:1 state1)
Using the Base Semantics given by fUML for Verification

Verification - Syntax

(A)
- \( c \in \mathbb{C} \) set of Classifiers
- \( z \in \mathbb{Z} \) set of Integers
- \( a \in \mathbb{A} \)
- \( an \in \mathbb{AN} \) set of ActivityNodes
- \( ae \in \mathbb{AE} \) set of ActivityEdges
- \( aan \in \mathbb{AAN} \)
- \( cn \in \mathbb{CN} \)
- \( en \in \mathbb{EN} \)
- \( on \in \mathbb{ON} \)
- \( ac \in \mathbb{ACN} \)
- \( fn \in \mathbb{FN} \)
- \( in \in \mathbb{IN} \)
- \( ffn \in \mathbb{FFN} \)
- \( owa \in \mathbb{OWA} \)
- \( sn \in \mathbb{SN} \) set of StructuredActivityNodes
- \( insn \in \mathbb{INSN} \)
- \( vsac \in \mathbb{VACN} \)
- \( pon \in \mathbb{PON} \)
- \( ipon \in \mathbb{IPON} \)
- \( opon \in \mathbb{OPON} \)
- \( aipin \in \mathbb{AIPON} \)
- \( aopin \in \mathbb{AOPON} \)
- \( sc \in \mathbb{SC} \)
- \( tg \in \mathbb{TG} \)
- \( inc \in \mathbb{INC} \)
- \( out \in \mathbb{OUT} \)
- \( rvzac \in \mathbb{RVACN} \)
- \( vsac \in \mathbb{VVACN} \)
- \( vs \in \mathbb{VS} \)
- \( lv \in \mathbb{LVS} \)
- \( vlv \in \mathbb{VLVS} \)

(buml:Classifier c)
(buml:Activity a)
(buml:ActivityNode an)
(buml:ActivityEdge ae)
(buml:activity an a)
(buml:ControlNode cn)
(buml:ExecutableNode en)
(buml:ObjectNode on)
(buml:Action ac)
(buml:ForkNode fn)
(buml:InitialNode fn)
(buml:FlowFinalNode ffn)
(buml:OWA = \( \mathbb{A} \subseteq (\mathbb{ACN} \cup \mathbb{ON}) \))

(buml:InStructuredNode an sn)
(buml:ValueSpecificationAction vsac)
(buml:Pin pon)
(buml:InputPin ipon)
(buml:OutputPin opon)
(buml:Input ac ipin)
(buml:Output ac opin)
(buml:Source ae an)
(buml:Target ae an)
(buml:Incoming an ae)
(buml:Outgoing ae an)
(buml:Result vsac opin)
(buml:Value vsac vs)
(buml:Value vs z)

(B)

Package AbstractSyntax

Activity

ActivityEdge

ObjectFlow

ExecutableNode

ControlNode

Pin

InitialNode

FinalNode

<<data Type>>

Integer

OutputPin

ValueSpecificationAction

LiteralInteger

LiteralSpecification

ValueSpecification

ControlFlow

ForkNode

ObjectNode

Event

In

Out

+value

result

+value

+value

+value

+value
Using the Base Semantics given by fUML for Verification

**Verification - Semantics**

(F)

[Action]

\[
\begin{align*}
& \forall (n.a) \ (\text{if} \ (\text{and} \ (\text{buml:activity} \ n.a) \\
& \text{\quad (form:executable-without-input} \ n.a)) < \\
& \text{\quad (forall} \ \text{obji f) } \\
& \text{\quad (if} \ (\text{form:classifies} \ \text{obji f) } \\
& \text{\quad (exists} \ \text{objin}) \\
& \text{\quad (form:property-value} \ \text{obji n objin f}))))) \Rightarrow s[PRV[\text{obji} \sqsubseteq n.s \rightarrow \text{objin}] \Rightarrow s']
\end{align*}
\]

(G)

[Action\(\text{vs}\)]

\[
\begin{align*}
& \forall (ac \ a vs opon) \ (\text{if} \ (\text{and} \ (\text{buml:ValueSpecificationAction} \ ac) \\
& \text{\quad (form:activity} \ ac a) \ (\text{buml:value} \ ac vs) \\
& \text{\quad (buml:result} \ ac opon)) \ (\text{forall} \ \text{objiac f) } \\
& \text{\quad (if} \ (\text{and} \ (\text{form:classifies} \ \text{obji f) } \\
& \text{\quad (form:property-value} \ \text{obji ac objiac f}))< \\
& \text{\quad (exists} \ \text{fxac}) \ (\text{and} \ (\text{form:holdsA} \ \text{fxac obji}) \\
& \text{\quad (forall} \ \text{v}) \ (\text{if} \ (\text{buml:value} \ vs \ v) \\
& \text{\quad (form:property-value} \ \text{obji opon v fxac}))))) \Rightarrow <\text{Actionvs}(ac), s> \Rightarrow s[PRV[\text{obji} \sqsubseteq opon]s \rightarrow ((ac \sqsubseteq \text{vilvs}), \sqsubseteq z)]
\end{align*}
\]
Verification – Theorem proving

**Verification goal:** for all executions of activity Main, the property variableX has the value 1.

(cl-text activityMainAlwaysRunValueSpecification1AndValueIsForm1
  (forall (exa f)
    (if (form:classifies Main exa f)
      (exists (f2)
        (form:property-value exa Main.ValueSpecificationAction1.variableX form:1 f2)
      )
    )
  )
)%implied
Agenda

• Introduction
  ▪ Motivation
  ▪ Goal

• Context

• Verification
  ▪ Example
  ▪ Syntax
  ▪ Semantics
  ▪ Theorem proving

• Conclusions
Conclusions

• Does fUML allow the use of model-checking or theorem proving?

• **Yes**, if it is used bUML. However,
  1. There is no standard way to go from bUML to fUML;
  2. The base semantics is in the process of maturation;
Conclusions - additional contribution

Maturation’s process - fUML (1.1 RTF beta)

• **Importance:** fUML is a basic building block for next specifications from OMG. For example: Precise Semantics for Composite Structures: “**new axioms** must have explicit relationships with the **base semantics**, and must be **consistent** with it.”

• but

  • a theorem prover (Eprover) was used to prove that base semantics is **not consistent**;
  • 42 **issues** were submitted to OMG;
  • 5 **enhancements** were proposed, e.g. refinement of **scope of bUML**.
Acknowledgment
USING THE BASE SEMANTICS GIVEN BY FUML FOR VERIFICATION

Presenter: Yu Bai
bai@cs.uni-kl.de

Alessandro Gerlinger Romero
romgerale@yahoo.com.br
Klaus Schneider
Maurício Gonçalves Vieira Ferreira
Brazilian National Institute for Space Research – INPE – Brazil
Department of Computer Science – University of Kaiserslautern – Germany

International Conference on Model-Driven Engineering and Software Development, MODELSWARD 2014
Lisbon, Portugal, January 7-9, 2014