Defining and Verifying Behaviour of Domain Specific Language with fUML

Qinan Lai and Andy Carpenter
School of Computer Science
The University of Manchester

BM-FA '12, July 03 2012, Kgs. Lyngby, Denmark.
Structure

• Backgrounds
  – Domain Specific Language (DSL)
  – fUML and ALF

• Framework
  – Achievement
  – Implementation
  – Example

• Further work
A Domain Specific Language (DSL) is a computer programming language focused on a particular domain.
Domain Specific Language

• Widely used in Model-driven development

• Syntax
  – Concrete syntax
  – Abstract syntax

• Behavioural Semantics
  – The behaviour of the DSL
  – The instructions of how to execute DSL programs
Define Behavioural Semantics

• UML as an example:
  – Abstract syntax: MOF
  – Behavioural semantics: English

• In reality
  – Abstract syntax: usually a meta-model
  – Behavioural semantics: can be anything!
Semantics specification of DSL

- Text
- State Machine
- Mathematics
- General Purpose Language
- Ontology
When creating semantics

• What *properties* my specification language should have?
  – Comprehensive specification
  – How large is the implementation
  – Tools

• How can I know that my semantics is *correct*?
  – No error (e.g. inconsistency, spelling error)
  – The semantics represents the correct execution instructions
Language for defining semantics

• Text leads to many problems
  – Ambiguity
  – Tool generation

• Formal languages
  – High learning curve
  – Or verbose
Existing approaches

• Translational approaches
  – Mapping: DSL -> well-defined semantic domain
  – Additional learning curve

• Weaving behaviour
  – Action Language
  – Intuitive for programmers
  – Non standard language (Kermeta[3] and XOCL[4])
Foundational subset of UML (fUML) [1]

- UML lacks of formal semantics specification
- Giving UML execution semantics
  - Kernel
  - CommonBehavior
  - Activities
- Semantic basis of UML
- Models are code. Models are deliverables.
Action Language for FUML (ALF)[2]

- Graphical modelling is not good for details
- ALF is the textual representation of UML
- Java-like syntax.
- Support a richer vocabulary and UML concepts.
More than Java

```java
//@parallel
{
startB();
startC();
}
Concurrent block

transitions -> select e (e.isActive()).at(1);
OCL-like sequence expression

accept(sig:SignalType);
Signal receive

if (condition1){
  //statements
}
or if (condition2){
  //statements
}
Concurrent if statement

Semantics of ALF is defined by mapping ALF concepts to fUML
Semantics definition based on fUML
Highlights

• A new way for defining semantics of DSL based on OMG standards.
  – Rich vocabulary
  – Combination of graphical and textual syntax

• Tools for supporting the framework
  – Editor
  – Compiler

• Verification on semantics specification
Tools for ALF

• Editor: Xtext
  – Aim at the full conformance level.
  – Support most expressions and statements

• Compiler: ATL transformation
Compile ALF code to fUML model
Petri Net Example
Graphical definition
@class=Net
activity runUntilDead(){
    while (transitions->exists e (e.isActive())){
        step();
    }
}

@class=Net
activity getActiveTransition():Transition{
    return transitions ->select e (e.isActive()).at(1);
}

@class=Place
activity product(){
    token = token + 1;
}

@class=Place
activity consume(){
    token = token - 1;
}

@class=Transition
activity isActive(): boolean{
    return src->forall e (e.token>0);
}
Semantics Verification

- Much research on UML activity diagram verification.
- Verify the semantics specification
- Static code verification
  - Adding OCL constraints on the behaviour models.
- Plan: dynamic verification
  - Deadlock checking
  - Executability
Execute a DSL program

Testing the semantics specification while creating it (rather than when implementing it).
Further work

• Dynamic verification
• More convincing example: BPEL case study
QUESTIONS?
Reference


