Model Transformation Semantic Analysis by Transformation

K. Lano, S. Kolahdouz-Rahimi, S. Yassipour-Tehrani

Dept of Informatics, King's College London, UK;

Dept of Software Engineering, University of Isfahan, Iran

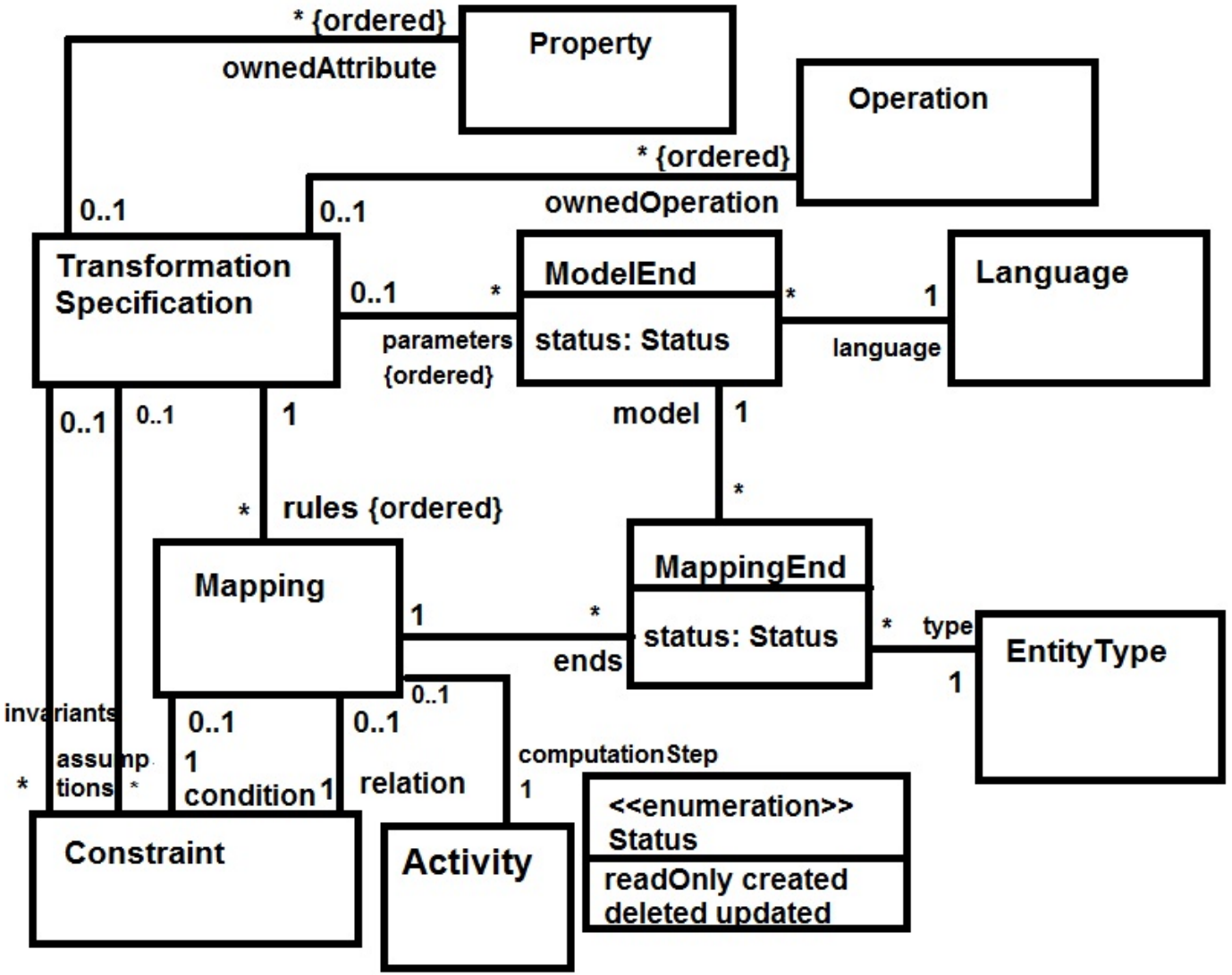
July 16, 2015

We describe a translation from QVT-R to an intermediate semantic representation, and use this to support analysis of QVT-R specifications.

The UML to RDB case study is used as an example.

Introduction

- MT verification hindered by large number of different MT languages.
- Our approach uses transformation metamodel $\mathcal{TMM}$ to express semantics of transformations in ATL, Flock, ETL, QVT-R, etc.
- QVT-R analysis: implicit deletion, change propagation and in-place execution.
- Translation to $\mathcal{TMM}$ provides logical semantics for QVT-R – supports proof of confluence and syntactic correctness.



Transformation specification metamodel $\mathcal{TMM}$

Analysis of QVT-R: UML2RDB example

QVT-R transformations specified by rules (`top relations` and non-top `relations`).

```
transformation tau(uml1 : SimpleUML, rdb1 : SimpleRDMS) {
  top relation Class2Table
  { checkonly domain uml1 c : Class {name = n};
    enforce domain rdb1 t : Table {name = n};
    where { Attribute2Column(c,t) }
  }

  relation Attribute2Column
  { checkonly domain uml1 c : Class { attribute =
      a : Attribute {name = an, type = typ:Type { name = tn }}};
    enforce domain rdb1 t : Table { column =
      col : Column {name = an, coltype = tn} };
  }
}
```

QVT-R element	*$\mathcal{TMM}$ representation*
RelationalTransformation	TransformationSpecification
modelParameter : Sequence(TypedModel)	parameters : Sequence(ModelEnd)
Relation (isTopLevel = true)	Mapping
Relation (isTopLevel = false)	Owned operation of TransformationSpecification
Relation variable	Variable defined in Mapping condition
RelationDomain (isEnforceable = false)	Mapping condition and readOnly MappingEnd
RelationDomain (isEnforceable = true)	Mapping relation and updated MappingEnd
Check-before-enforce semantics	Unique instantiation semantics
Key	Identity attribute

Interpretation of top relations

A top relation r is interpreted as Mapping Con_r with *readOnly* ends for each checkonly domain of r, and *updated* ends for each enforce domain of r.

when clause is interpreted as condition predicate; *where* clause interpreted as relation predicate *wherep*.

Creation of trace object for rule application also included in relation.

Eg., interpretation of $Class2Table$ as Mapping $Con_{Class2Table}$:

$$
\begin{aligned}
&c : Class \ \&\ n = c.name \ \Rightarrow \\
&\quad Table \rightarrow exists(t \mid t.name = n \ \&\ attribute2column\$op(c,t) \ \& \\
&\qquad Class2Table\$trace \rightarrow exists(class2table \mid \\
&\qquad\quad class2table.c = c \ \&\ class2table.t = t))
\end{aligned}
$$

QVT-R transformation τ is interpreted as TransformationSpecification with rules Con_r for top-level *Relations* r of τ.

For non-top-level relations r, a predicate Op_r is derived as postcondition of an update operation op_r. Eg.:

```
relation Attribute2Column
{ checkonly domain uml1 a : Attribute
                { type = t : Type { name = tn }};
  enforce domain rdb1 c : Column { coltype = tn };
}
```

has interpretation as operation

$$attribute2column\$op(a : Attribute, c : Column)$$
$$post: \ t = a.type \ \& \ tn = t.name \ \Rightarrow \ c.coltype = tn \ \&$$
$$Attribute2Column\$trace \rightarrow exists(attribute2column \mid$$
$$attribute2column.a = a \ \& \ attribute2column.c = c)$$

Expressing implicit deletion

QVT-R has semantic feature of *implicit deletion*: target elements are deleted if no rule requires them to exist.

Expressed semantically by transformation *invariants*: constraints which hold at start and throughout transformation execution.

The invariants Inv_r express that target model elements can be derived from source model elements via top-level relation r.

$Inv_{Class2Table}$ is:

$$t : Table \ \& \ n = t.name \ \Rightarrow \\ Class \rightarrow exists(c \mid c.name = n \ \& \ attribute2column\$inv(c, t))$$

If condition P: $n = t.name$ holds, but succedent Q does not hold, then t should be deleted.

Analysis techniques

After translation to $\mathcal{TMM}$, verification can be carried out:

- Definedness and determinacy
- Analysis of data dependencies
- Invariants, syntactic correctness, model-level semantic preservation
- Counter-example executions.

Class2Table identified as confluent and type 1 (its write frame is disjoint from read frame).

Would be error to write:

```
relation Attribute2Column
{ checkonly domain uml1 c : Class { name = n,  attribute =
    a : Attribute { name = an, type = typ : Type { name = tn }};
  enforce domain rdb1 t : Table { name = n, column = col : Column
     { name = an + t.column->size(), coltype = tn }};
}
```

This version both reads and writes *Table* :: *column* – warning would be issued.

UML-RSDS tool output:

```
Analysing use case: tau

WRITE FRAME of constraint 0 =
  {Table::name, Table::column, Column::name, Column, Table}
READ FRAME of constraint 0 =
  {UMLClass, UMLClass::name, UMLClass::attribute,
   Attribute::name, Table::column}

Type 2 constraint: writes and reads {Table::column}
Fixpoint iteration needed. Proof using a variant
needed for confluence & termination
```

In-place transformations

- A single model is both read and updated.
- Semantic translation to $\mathcal{TMM}$ can be used to give semantics for in-place QVT-R specifications, but with fixed-point execution of computationStep.
- Implicit deletion expressed by 'default deletion' rules: if no normal rule creates or copies an element, then element is deleted.

Conclusions

Techniques provide language-independent MT verification capabilities via translations to common semantic representation.

Translations to $\mathcal{TMM}$ also implemented for ATL, Flock and ETL.

Approach can be extended in principle to TGG, GrGen.NET and QVT-O.