STAF WORKSHOP ON VERIFICATION OF MODEL TRANSFORMATIONS
L'AQUILA, ITALY. JULY 23, 2015
VOLT 2015

Verification of the CD2RDBMS Transformation Case in Flora-2.

Muzaffar Igamberdiev, Georg Grossmann, and Markus Stumptner

Advanced Computing Research Centre,
School of IT and Mathematical Sciences
University of South Australia,
Mawson Lakes, SA 5095, Australia

http://kse.cis.unisa.edu.au/

KSE Lab
Knowledge and Software Engineering Laboratory

University of South Australia

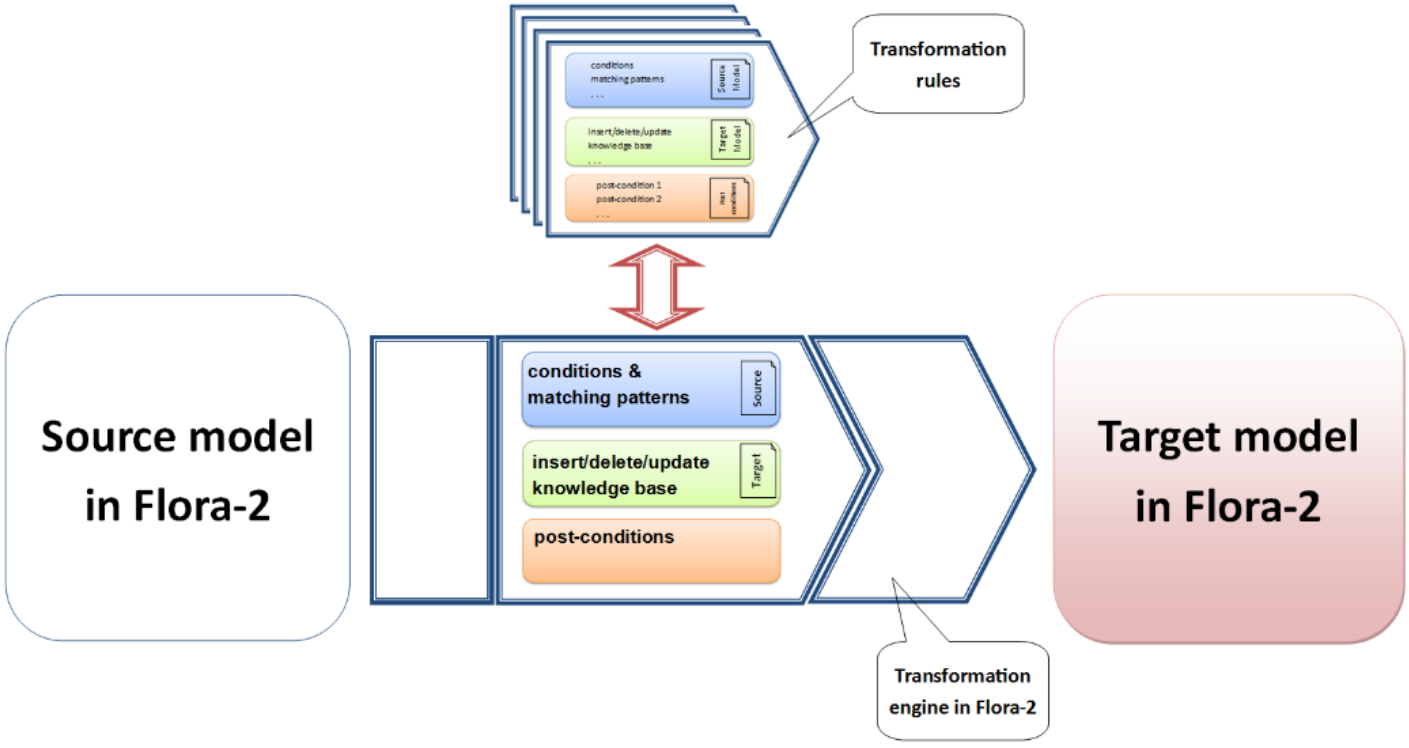
A Single language framework - **MOTIF**

Modeling, transformation and verification in Flora-2

Why is it a good idea ?

- Integrating modeling and verification into a single framework
- A single language for modeling, transformation and verification
- Reasoning features of Flora-2 open-source language
- Verification feedback within the framework
- Object-oriented – easier to map

Transformation & verification framework




```
%TransformClass2Table(?CLASS, ?TABLE, ?SrcModule, ?TargetModule) :-
 ((?CLASS[is_persistent->true,name->?_NAME]:class@?SrcModule;
 \+(?CLASS :: ?_Y))),
 insert{(?TABLE[name->?_NAME] : table)@?TargetModule}.
```

Implementation

Modeling – representation/encoding in Flora-2

```
Classifier[name*=>_string].
PrimitiveDataType :: Classifier.
Class :: Classifier.
Association[name*=>_string].
Association[src*=>Class].
Association[dest*=>Class].
Class[is_persistent*=>_boolean].
Class[parent*=>Class].
Class[attrs{0:*}*=>Attribute].
Attribute[is_primary:_boolean].
Attribute[name*=>_string].
Attribute[type*=>PrimitiveDataType].
```

```
Table[name*=>_string].
Table[fkeys{0:*}*=>FKey].
Table[pkey{0:*}*=>Column].
Table[cols{0:*}*=>Column].
FKey[references*=>Table].
FKey[cols{0:*}*=>Column].
Column[type*=>_string].
Column[name*=>_string].
```

Implementation

Verification rules are defined in Flora-2 rules

1. Non-persistent classes and non-top classes must not be transformed into a corresponding table.

```
non_persistent_non_top_classes(?C, ?T, ?SrcM, ?TargetM ):-
        ?C[is_persistent->false] : Class@?SrcM,
        ?C[parent->?_Y]@?SrcM,
        ?C[name->?_CNAME]@?SrcM,
        ?T[name->?_CNAME] : Table@?TargetM,
```

2. All persistent top classes must be transformed into a corresponding table

```
persistent_top_classes(?C, ?SrcM, ?TargetM) :-
        ?C[is_persistent->true] : Class@?SrcM,
        \+(?C[parent->?_Y])@?SrcM,
        ?C[name->?_CNAME]@?SrcM,
        forall(?_T)^(\+(?_T[name->?_CNAME]) : Table@?TargetM).
```

```
flora2 ?- persistent_top_classes(?C,main,main).
?C = a
1 solution(s) in 0.0000 seconds
Yes
```

Implementation

Verification rules are defined in Flora-2 rules

3. Column duplicates are forbidden in the output models, i.e., there should not be two columns with the same name in one table.

```
no_column_duplicates(?T,?TargetModule) :-
        ?T : Table[cols->?COL1,cols->?COL2]@?TargetModule,
        ?COL1 \= COL2,
        ?COL1[name->?_C1]@?TargetModule,
        ?COL2[name->?_C1]@?TargetModule.
```

4. Querying transformation rules in transformation specification

```
?- clause{%TransClass2Table(?_,?_),((?_:Class@?_, ?X), ?Y)}.

?X = (${?_h5888[is_persistent->true]@_h5886},
${?_h5888[name->?_h5913]@_h5886})
?Y = ${insert{?_h5940[name->?_h5913]@_h5938?_h5940:Table@_h5938}}

1 solution(s) in 0.0000 seconds
```

Model transformation verification approaches

Approach	Correctness properties	MT language	Verification approach /language	A single language framework
UMLtoCSP constraint programming [4,3]	satisfiability, lack of constraint redundancies & subsumptions, liveliness	UML/OCL to Constraint Satisfaction Problem (CSP)	ECLiPSe constraint programming system	NO
UML/OCL Boolean satisfiability [15]	consistency of system states & redundancy of OCL constraints	UML models, OCL >SAT instances	SAT solver	NO
CARE [14]	conformance	Xtend	Answer Set Programming(ASP)	NO
Language independent MT verification [11]	termination, single inheritance, name conflicts and others	Transformation spec meta-model is applied on ATL	an intermediate represent, lang independent framework	NO
UML/OCL Validator [7]	transf model consistency, property preservation	transformation models	USE model validator	NO
MOTIF	conformance, completeness & inconsistencies	Flora-2	Flora-2	YES

Conclusion

- A single language framework – **MOTIF** – for modeling, transformation and verification.
- Verification based on the source and the target (meta-) models
- Verification on rule base
- Design-time (pre-transformation) verification of transformation rules

THANK YOU!

Implementation

Verification rules are defined in Flora-2 rules

```
unique_table_name(?T) :-
        ?T[name->?_N1] : Table,
        ?_T[name->?_N2] : Table,
        ?_N1 = ?_N2.
```

```
unique_artifact_name(?T, ?A) :-
        ?T[name->?_N1] : ?A,
        ?_T[name->?_N2] : ?A,
        ?_N1 = ?_N2.
```

```
?- ?errors = setof {?_error |
        unique_artifact_name(?_T,?_A),
        ?_error=[?_T,?_A]},
        ?errors.length@_basetype=?N_violations.
```