“The Implications of Optimality Results for Incremental Model Synchronization for TGGs”

Bi-directional transformations (BX) – Theory and Applications Across Disciplines (13w5115) December 1 - 6, 2013

Holger Giese, Stephan Hildebrandt
System Analysis & Modeling Group,
Hasso Plattner Institute for Software Systems Engineering
University of Potsdam, Germany
holger.giese@hpi.uni-potsdam.de
Outline

1. Introduction
2. Optimality Result
3. Open Issues & Implications
1. Introduction: Triple Graph Grammars

**Problem:**
- Describe full model transformations where source and target can have different meta-models

**Solution:**
- TGGs: describe relation between source and target by means of a grammar that generates all valid pairs (here even triples incl. correspondence)
- Derive operational rules in form of Story Pattern for
  - forward direction,
  - backward direction, and
  - consistency
1. Introduction: Theoretical Results on TGGs

- MDELab is a toolset for our Triple Graph Grammar dialect (and Story Diagrams) covering also testing and analysis [1]

**Consistency, Completeness of TGG:**
- Well-formedness rules which guarantee consistency and completeness for our TGG dialect [2] ➔ apply to model synchronization as well

**Testing of TGGs:**
- Automated test generation for our TGG dialect offering a complete coverage of the TGG specifications [3,4] ➔ apply to model synchronization as well

**Verification of TGGs (and related GTS):**
- Automatic checking behavior preservation for our TGG dialect [5] ➔ apply to model synchronization as well
- Automatic checking of structural constraints for transformation rules [6] ➔ apply to TGGs and model synchronization as well
1. Introduction: Practical Results on TGGs

Applications:
- Industrial Case Study on the Integration of SysML and AUTOSAR with Triple Graph Grammars via model synchronization [7,8]
- Application of model synchronization for Runtime Models [9]

Efficiency:
- Improved performance for Story Diagrams using runtime information [12]
- Optimal performance for incremental model synchronization for our TGG dialect for practically relevant models [13,14] based on the results for Story Diagrams [12]

Why is it relevant, what is the result, and why does it work?
2. Optimality Result: Motivation

Why does performance matter at all?

- Reducing the effort also means to minimize the unnecessary changes; **optimal means no unnecessary changes at all!**

- We may have multiple model synchronizations linked together additional steps such as consistency checks, and/or rollbacks and thus the overall performance matters (model management)

- Short execution times make conflicts very unlikely (if models are handled by the same tool or a shared repository)

- Besides case tools we apply model synchronizations also for runtime models where the overhead should be minimal
2. Optimality Result: Synchronization Problem

- Model Synchronization

- Incremental Model Synchronization

**Remark:** concurrent updates are not considered!
2. Optimality Result:  
Synchronization Problem

- forward: $\Delta_s$ implies to compute $\Delta_c + \Delta_T$
- backward: $\Delta_T$ implies to compute $\Delta_c + \Delta_s$  
  (can be handled analogous to the forward case)

Remark: $O(\Delta_s + \Delta_c + \Delta_T)$ is thus a lower complexity bound
2. Optimality Result: Restrictions of our TGG dialect

The rules of our TGG dialect allow only to link finite many elements in the source and target model to each other via one correspondence elements at once. Consequently,

- $O(\Delta_C) = O(\Delta_S + \Delta_C + \Delta_T)$
- $O(\Delta_C)$ is also a lower bound for the synchronization problem

Implications for BX:
- Do similar restrictions apply for other techniques (GT or other domains)?
2. Optimality Result: Practically Relevant Models

Study of available data concerning model transformations reveals:

- P1 The models contain at most one “unbounded” link (link in $O(n)$-$O(1)$).
- P2 Source and target models are weakly connected graphs.
- P3 Expressions in the rules can be evaluated in constant time.
- P4 TGG transform all elements of the source and target models*

Implications for BX:
- Do similar limitations for practical cases also apply in other domains (SE is already considered)?
2. Optimality Result: Optimal Model Synchronization

- synchronization algorithm with repair ➔ The overall synchronization requires $O(\Delta_C)$ rule checks and applications.

- dynamic pattern matching strategy ➔ Finding all matches for a single rule starting from a given correspondence is in $O(\Delta_C)$ steps, if it conceptually has to traverse the unbounded link (by using the untransformed elements set) or $O(1)$ otherwise.

- As for one correspondence only for each rule all matches have to be searched (still $O(\Delta_C)$), the combined execution of the checks and applications for the synchronization scheme requires also only $O(\Delta_C)$ steps.

Result: if P1-P4 hold, complexity is $O(\Delta_C)$ and thus **optimal**!
2. Optimality Result: Results for the Implementation

- UML2RDBMS standard benchmark
- org.eclipse.emf reduced: 3227 classes and 2295 associations
- org.eclipse.emf: 6455 classes and 6761 associations

→ no increase with growing model size!
2. Optimality Result: Results for the Implementation

- UML2RDBMS standard benchmark
- generated test data ...

→ linear increase with growing delta size!

BX 2013 | Holger Giese | The Implications of Optimality Results for Incremental Model Synchronization
2. Optimality Result: Results for the Implementation

- SDL2UML example with adding 1000 elements
- UML model is about six times the number of rule applications
  ➔ some minimal increase with growing model size due to EMF!

BUT EMF does not scale as well …
3. Open Issues & Implications

Open Issues (for TGGs):
- Which not yet supported TGG concepts can be added without loosing the optimality result? Priorities ✓, ... ?

Implications (for TGGs):
- Multiple model synchronization with TGGs that link multiple models would still be highly efficient (remark: interpreter vs. compiler!)

Implications (for BX):
Observation: The restrictions for practical problems matter!
- Maybe it is more promising to look for solutions for relevant classes of practical problems rather then the general case?
- Maybe we can decompose relevant problems into some relevant classes and compose our solution from solutions for each class?


