Algebraic Graph Transformation and Model Transformation

Fernando Orejas
Royal Holloway University of London
on leave from Universitat Politècnica de Catalunya, Barcelona
1. Introduction
2. Specification of Model Transformations by Triple Graph Grammars
3. Implementing Model Transformations
4. Synchronizing Model Transformations
Introduction
What is a model?
What is a model?
What is a model?
What is a model?
What is a model?
What is a model?
What is a model?
What is a model transformation?
What is a model transformation?
What is a model transformation?
What is a model transformation?
What is a model transformation?

A model transformation is a procedure that given instances of a certain source model class it returns instances of a target model class.

A "standard language": QVT
Kinds of Model Transformations:

• Monodirectional or Bidirectional
• Horizontal or Vertical
• Endogenous or exogenous
• From more abstract to more concrete
• From more concrete to more abstract
What is a model transformation?
What is a model transformation?
A model transformation is a set of triple graphs

\[ G_S \quad \xleftarrow{h_S} \quad G_C \quad \xrightarrow{h_T} \quad G_T \]
Triple graph morphisms:

\[
\begin{array}{c}
M_S & h_S & M_C & h_T & M_T \\
\downarrow g_S & & \downarrow g_C & & \downarrow g_T \\
M_S & \quad & h_S & \quad & h_T \\
\end{array}
\]
Problems

• Specification
• Implementation
• Synchronization
Specification of model
Transformations by Triple
Graph Grammars
(Many) Classes classes of model transformations can be naturally specified by triple graph grammars (TGGs).
A TGG consists of:
A starting triple graph
A triple graph transformation rule is:

$$p = L \rightarrow R$$
Transformation Rules

class-table creation
attribute-column creation
Subclass creation
The model transformations specified by a TGG is the class of triple graphs that can be generated by the grammar.
Rule application
Class-Table creation
Class-Table creation

Subclass creation
Class-Table creation

Subclass creation

Attribute-column creation
Compact Notation of Transformation Rules
Implementation
The problem:
Given a TGG rule

we can obtain the following operational rules
Source Rule

\[ L_S \leftrightarrow \emptyset \rightarrow \emptyset \]

\[ R_S \leftrightarrow \emptyset \rightarrow \emptyset \]
Rule

attribute-column creation
Source Rule

attribute-column creation
Given a TGG rule
Forward Rule
Rule

attribute-column creation
Forward rule

attribute-column creation
Fact:

\[ G \xrightarrow{p} G' \]

If and only if

\[ G \xrightarrow{p_S} G'' \xrightarrow{p_F} G' \]
Implementing Model Transformations:

\( (\emptyset, \emptyset, \emptyset) \xrightarrow{} p_1 G_1 \xrightarrow{} p_2 \xrightarrow{} \ldots \xrightarrow{} p_n G \)

If and only if

\( (\emptyset, \emptyset, \emptyset) \xrightarrow{} p_1 S G'_1 \xrightarrow{} p_1 F G_1 \xrightarrow{} p_n S \xrightarrow{} \ldots \xrightarrow{} G_{n-1} \xrightarrow{} p_1 S G'_n \xrightarrow{} p_n F G \)

If and only if

\( (\emptyset, \emptyset, \emptyset) \xrightarrow{} p_1 S G''_1 \xrightarrow{} p_2 S \xrightarrow{} \ldots \xrightarrow{} p_n S G' \xrightarrow{} p_1 F G'_1 \xrightarrow{} p_2 F \xrightarrow{} \ldots \xrightarrow{} p_n F G \)
Example

Rule 1^s

Rule 3^s

Rule 2^s
Example

Rule 1$^F$

Rule 3$^F$

Rule 2$^F$
Consistency of Forward Transformation Sequences

A forward transformation sequence

\[ G \xrightarrow{p_1^F} G_1 \xrightarrow{p_2^F} \ldots \xrightarrow{p_n^F} G' \]

is source consistent if:

\[ (\emptyset, \emptyset, \emptyset) \xrightarrow{p_1^S} G_1' \xrightarrow{p_2^S} \ldots \xrightarrow{p_n^S} G \]
Theorem

Source consistent forward transformation sequences are correct and complete
Problem

How to compute efficiently source consistent forward transformation sequences.

Answer

Using *forward translation rules*

Basic idea

Find the source derivation while constructing the forward derivation, by simulating the construction of the source model.
Forward Translation Rules

All elements have a boolean translation attribute, meaning if the element *has been created* or not.

Forward translation rules are like forward rules but, on the left-hand side:

- The translation attribute of elements in $R_S \setminus L_S$ is false.
- The translation attribute of elements in $L_S$ is true.

And, on the right hand side:

The translation attribute of all elements in $R_S$ is True.
A Forward Translation Rule

attribute-column creation
Applying Forward Translation Rules

Given a source model $G_S$, its associated triple graph with translation attributes is $\langle F(G_S), \emptyset, \emptyset \rangle$, where $F(G_S)$, is $G_S$ with all the translation attributes set to False.

A forward translation sequence:

$$(F(G_S), \emptyset, \emptyset) \xrightarrow{p_1 FT} \ldots \xrightarrow{p_n FT} (G'_S, H_C, H_T)$$

is complete if all translation attributes of $G'_S$ are true.
Theorem

Complete translation sequences are correct and complete.

Moreover, if all rules in the given TGG create some source element, then translation sequences are finite.
The image consists of a series of diagrams showing a sequence of transformations involving variables C, T, and A. The diagrams depict logical or state transitions, with arrows indicating changes between states. The process starts with three panels on the left, each showing a state for C, T, and A with values of F (false) or T (true). The sequence of transformations progresses through several steps, indicated by the rightward arrows, leading to a final panel on the right showing a new state for C and A, with T and Co, suggesting a completion or a new phase in the sequence.
Conflict detection and elimination

We can detect conflicts using critical pairs.
attribute-column creation
Subclass creation
It is not a critical pair
Transformation Rules

class-table creation
Subclass creation
Critical Pair

\[\text{:C} \xrightleftharpoons{} \text{:T} \]
Conflict detection and elimination

We can eliminate conflicts using *filter NACs.*
Filter NACs
Theorem

Filter NACs preserve completeness.
Model Synchronización
The problem
The problem
The problem
Transformation rules

:Person

:Person Dept: Tech.

:Person
:Name

:Person
:Name
Transformation rules

\[ \text{:Person} \quad X:\text{Base} \quad Y:\text{Bonus} \quad \text{\rightarrow} \quad \text{:Person} \quad X+Y:\text{Salary} \]

\[ \text{:Person} \quad \text{Dept: Manag.} \]
The problem
The problem
The solution

1. Alignement
Alignement

Bill Clinton
Base: 3500
Bonus: 1500
Melinda French
Base: 3000
Bonus: 1000

Bill Gates
Dept: Manag.
Salary: 6000

Melinda Gates
Base: 3000
Bonus: 2000

Bill Clinton
Dept: Tech.
Salary: 5000

Melinda French
Dept: Tech.
Salary: 4000
2. Deletion

Find maximal $G'^{0}_{S} \leftarrow G'^{0}_{C} \rightarrow G'^{0}_{T}$ included in $G'^{0}_{S} \leftarrow G'^{0}_{C} \rightarrow G'_T$.
Deletion

Bill Clinton
Base: 3500
Bonus: 1500

Melinda French
Base: 3000
Bonus: 1000

Bill Clinton
Dept: Tech.
Salary: 5000

Melinda French
Dept: Tech.
Salary: 4000

Bill Gates
Dept: Manag.
Salary: 6000

Melinda Gates
Base: 3000
Bonus: 2000

Bill Gates
Dept: Manag.
Salary: 6000

Melinda French
Dept: Tech.
Salary: 4000
3. Addition

Apply forward translation rules to obtain $G'_S \leftarrow G'_C \rightarrow G'_T$
Addition

Bill Clinton
Base: 3500
Bonus: 1500

Melinda French
Base: 3000
Bonus: 1000

Bill Clinton
Dept: Tech.
Salary: 5000

Melinda French
Dept: Tech.
Salary: 4000

Bill Gates
Dept: Manag.
Salary: 6000

Melinda Gates
Base: 3000
Bonus: 2000

Bill Gates
Dept: Manag.
Salary: 6000

Melinda Gates
Dept: Tech.
Salary: 5000
Theorem

Model Synchronization is correct, complete and hipocratic
Concurrent Synchronization
The Problem
The solution

1. Propagating source update

\[
\begin{array}{c}
G_S & \xrightarrow{} & G_T & \xrightarrow{} & G'_T \\
\downarrow & & \downarrow & & \downarrow \\
G'_S & \xleftarrow{} & G''_T & \xrightarrow{} & G'_T
\end{array}
\]
Propagating source update

Before update:

Bill Clinton  
Dept: Tech.  
Salary: 5000

Melinda French  
Dept: Tech.  
Salary: 4000

Bill Gates  
Dept: Manag.  
Salary: 6000

After update:

Bill Clinton  
Dept: Manag.  
Salary: 6000

Melinda French  
Dept: Tech.  
Salary: 4000

Bill Gates  
Dept: Manag.  
Salary: 6000

Melinda Gates  
Dept: Tech.  
Salary: 5000

Bill Gates  
Dept: Manag.  
Salary: 6000
2. Conflict Resolution
3. Propagating target update
Propagating target update

Bill Clinton
Base: 3500
Bonus: 1500

Melinda French
Base: 3000
Bonus: 1000

Bill Gates
Dept: Manag.
Salary: 6000

Melinda Gates
Dept: Tech.
Salary: 5000

Bill Clinton
Dept: Tech.
Salary: 5000

Melinda French
Dept: Tech.
Salary: 4000

Bill Gates
Dept: Manag.
Salary: 6000

Bill Gates
Dept: Manag.
Salary: 6000

Melinda Gates
Dept: Tech.
Salary: 5000

Melinda Gates
Base: 3000
Bonus: 2000
Theorem

Concurrent synchronization is correct and complete.
Some problems

- Attributed graphs
- Correctness criteria for concurrent synchronization
- Multi-party concurrent synchronization
¡Thank You!