

Model Transformation with Triple Graph Grammars and Non-terminal Symbols

William da Silva, Max Bureck, Ina Schieferdecker, and
Christian Hein

Fraunhofer Fokus, Berlin, Germany
`william.bombardelli.da.silva@fokus.fraunhofer.de`
Technische Universität Berlin, Berlin, Germany

November 2, 2018

Organization

- 1 Introduction
- 2 Triple Graph Grammars with Non-terminal Symbols
- 3 Evaluation
- 4 Conclusion
- 5 References

Introduction

- 1 Introduction
- 2 Triple Graph Grammars with Non-terminal Symbols
- 3 Evaluation
- 4 Conclusion
- 5 References

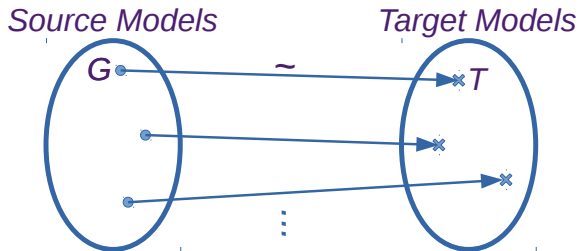
- Model-driven software development as a technique to enhance quality of software

- Model-driven software development as a technique to enhance quality of software
- Models as formal specifications of safety-critical systems

- Model-driven software development as a technique to enhance quality of software
- Models as formal specifications of safety-critical systems
- Transformation between models (e.g. from a formal specification to high-level source-code and vice-versa)

- Model-driven software development as a technique to enhance quality of software
- Models as formal specifications of safety-critical systems
- Transformation between models (e.g. from a formal specification to high-level source-code and vice-versa)
- **Goal:** Comprehensible and reliable transformations
 - Efficient representation of abstract concepts
 - Small size

The Model Transformation Problem



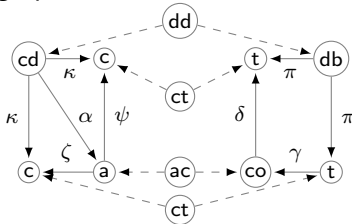
- $G \sim T$ iff G is correctly transformed into T
- \sim is the *correctly-transformed relation* between source and target models
- **Batch forward transformation:**
Given G , find a T , such that $G \sim T$

The Triple Graph Grammar Approach

- Models are graphs

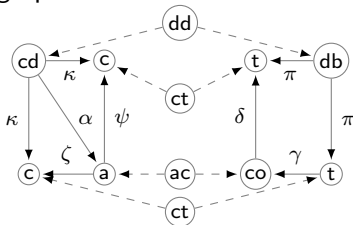
The Triple Graph Grammar Approach

- Models are graphs
- Two correctly-transformed graphs G and T are in a triple graph $G \leftarrow C \rightarrow T$



The Triple Graph Grammar Approach

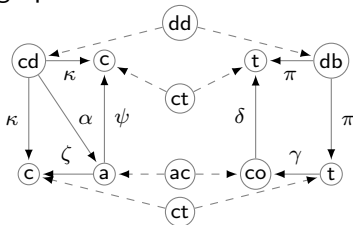
- Models are graphs
- Two correctly-transformed graphs G and T are in a triple graph $G \leftarrow C \rightarrow T$



- A triple graph grammar TGG is a generator of a set of triple graphs $L(TGG)$

The Triple Graph Grammar Approach

- Models are graphs
- Two correctly-transformed graphs G and T are in a triple graph $G \leftarrow C \rightarrow T$

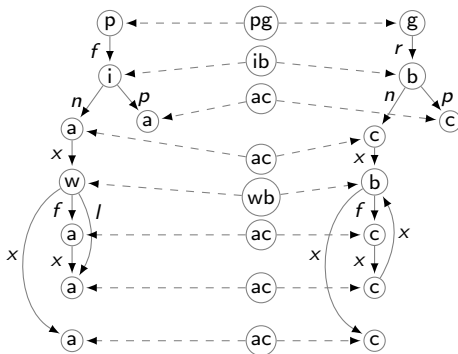


- A triple graph grammar TGG is a generator of a set of triple graphs $L(TGG)$
- The correctly-transformed relation \sim between graphs is described in terms of a triple graph grammar TGG
 - $G \sim T$ iff $(G \leftarrow C \rightarrow T) \in L(TGG)$

TGG – An Example

■ Pseudocode to Controlflow

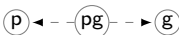
```
program main(n)  
if n < 0 then  
    return Nothing  
else  
    f ← 1  
    while n > 0 do  
        f ← f * n  
        n ← n - 1  
    end while  
    return Just f  
end if
```



TGG - An Example

■ Pseudocode to Controlflow

$r_0 :$

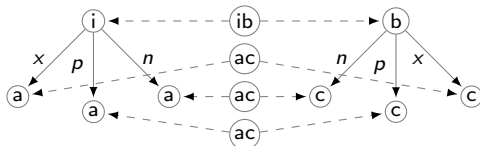
$::=$ 

$r_1 :$

 $::=$ 

$r_2 :$

 $::=$



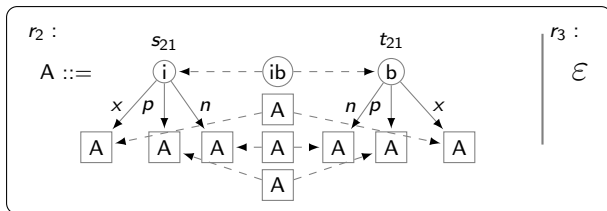
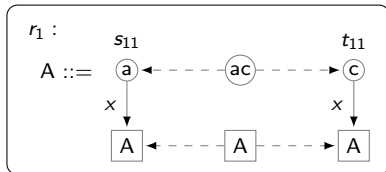
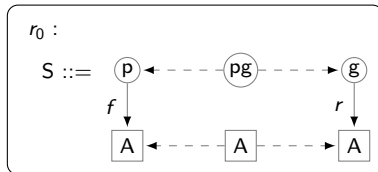
Triple Graph Grammars with Non-terminal Symbols

- 1 Introduction
- 2 Triple Graph Grammars with Non-terminal Symbols**
- 3 Evaluation
- 4 Conclusion
- 5 References

- New formalism: NCE TGG
 - *Graph Grammar with Neighborhood-controlled Embedding* (NCE) [Janssens and Rozenberg(1982)]
 - *Triple Graph Grammar* (TGG) [Schürr(1994)]
- Non-terminal symbols
- Context-free

NCE TGG – An Example

■ Pseudocode to Controlflow



- 1 Introduction
- 2 Triple Graph Grammars with Non-terminal Symbols
- 3 Evaluation**
- 4 Conclusion
- 5 References

Transformation	Standard TGG		BNCE TGG	
	Rules	Elements	Rules	Elements
Pseudocode2Controlflow	45	1061	7	185
BTree2XBTree	4	50	5	80
Star2Wheel	-	-	6	89
Class2Database	5	80	-	-

Table: Results of the usability evaluation of the BNCE TGG formalism in comparison with the standard TGG for the model transformation problem

Conclusion

- 1 Introduction
- 2 Triple Graph Grammars with Non-terminal Symbols
- 3 Evaluation
- 4 Conclusion**
- 5 References

- New context-free TGG formalism
 - Used to specify model transformations
 - Outperforms standard TGG in 2 evaluated cases
 - Special potential for code-generation
 - Cannot model important transformations (e.g. Class Diagrams)

- New context-free TGG formalism
 - Used to specify model transformations
 - Outperforms standard TGG in 2 evaluated cases
 - Special potential for code-generation
 - Cannot model important transformations (e.g. Class Diagrams)
- Future Work
 - Application conditions: Positive experimental results
 - Broader evaluation including empirical assessment with engineers and performance reports
 - Model synchronization

- 1 Introduction
- 2 Triple Graph Grammars with Non-terminal Symbols
- 3 Evaluation
- 4 Conclusion
- 5 References**



Dirk Janssens and Grzegorz Rozenberg.

Graph grammars with neighbourhood-controlled embedding.
Theoretical Computer Science, 21(1):55–74, 1982.



Andy Schürr.

Specification of graph translators with triple graph grammars.
In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 151–163. Springer, 1994.