

Transforming Process Algebra Models into UML State Machines: Bridging a Semantic Gap?

Marcel van Amstel Mark van den Brand Zvezdan Protić
Tom Verhoeff

Eindhoven University of Technology, The Netherlands

March 26, 2008

Presentation outline

- 1 Introduction
- 2 Transformation
- 3 Implementation
- 4 Illustration
- 5 Conclusions

Introduction

- FALCON: automated software generation from models in a warehouse environment
- Transformation from ACP (without ∂ operator) to UML state machines
- Model transformations encompass more than just transforming syntax

Transformation

Requirements

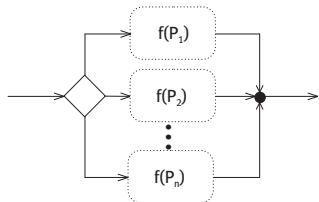
- Behavioral equivalence
- Structural equivalence with respect to parallel behavior

Transformation

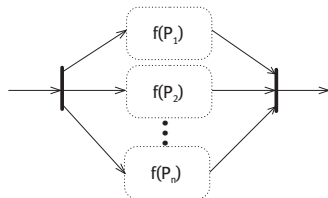
Mapping



Sequential composition (\cdot)



Alternative composition ($+$)



Parallel composition (\parallel)

Transformation

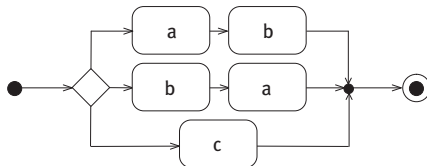
Semantic gap

- $a||b = a \cdot b + b \cdot a + a|b$
- ACP uses γ to determine whether a and b will communicate
- UML state machines do not have such a mechanism

Transformation

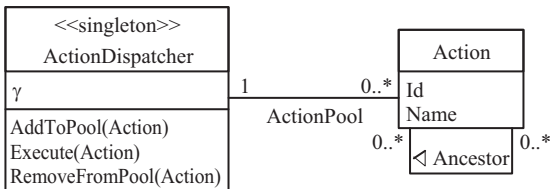
Rewriting

Suppose : $a \parallel b$, and $\gamma(a, b) = c$



Transformation

Action dispatcher



Transformation

Mapping continued



Atoms (actions, δ , ϵ)

Transformation

Action dispatcher continued

1. *AddToPool*(x : Action):
2. ActionPool := ActionPool \cup { x };
3. $\forall_a : \gamma(a.Name, x.Name) = y$
4. \rightarrow if $a \in$ ActionPool $\wedge a.Id \notin x.Ancestor$
5. \rightarrow NewAction := Action(NewId(), y , $a.Ancestor \cup x.Ancestor \cup \{a, x\}$);
6. *AddToPool*(NewAction)

7. *Execute*(x : Action):
8. $\forall_a : a.Id \in x.Ancestor$
9. \rightarrow *Execute*(a)
10. *RemoveFromPool*(x)

11. *RemoveFromPool*(x : Action):
12. ActionPool := ActionPool $-$ { x };
13. $\forall_a : a \in$ ActionPool
14. \rightarrow if $x.Id \in a.Ancestor$
15. \rightarrow *RemoveFromPool*(a)

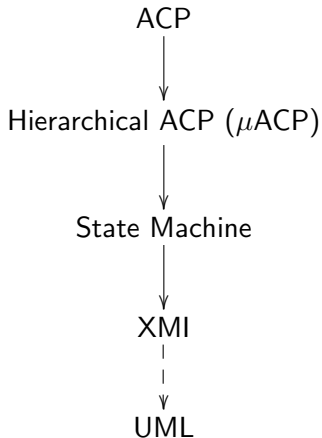
Implementation

Transformation scheme



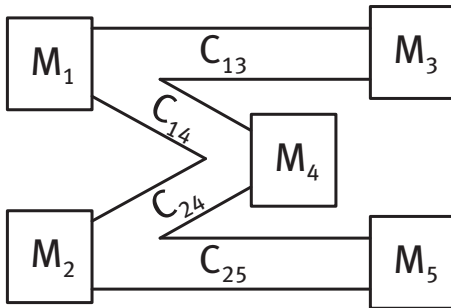
Implementation

Transformation scheme



Illustration

Conveyor system



Illustration

ACP model

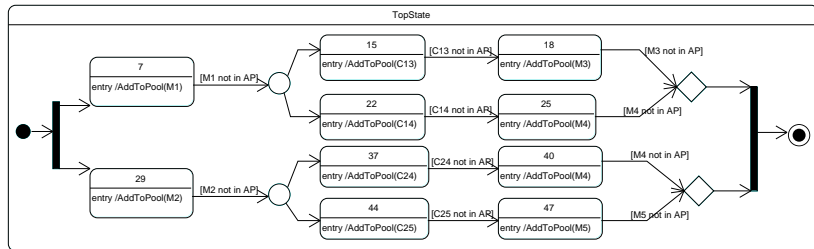
$$\gamma(C_{14}, C_{24}) = \textit{operator}$$

$$M_1 \cdot (C_{13} \cdot M_3 + C_{14} \cdot M_4) \quad \parallel \quad M_2 \cdot (C_{25} \cdot M_5 + C_{24} \cdot M_4)$$

Illustration

State machine

$$M_1 \cdot (C_{13} \cdot M_3 + C_{14} \cdot M_4) \quad || \quad M_2 \cdot (C_{25} \cdot M_5 + C_{24} \cdot M_4)$$



Illustration

State machine simulation

```
Executable started  
Dispatcher Started  
Executing : M1  
Executing : M2  
Executing : C24  
Executing : C13  
Executing : M4  
Executing : M3
```

```
Executable started  
Dispatcher Started  
Executing : M1  
Executing : M2  
Executing : operator  
Executing : M4  
Executing : M4
```

```
Executable started  
Dispatcher Started  
Executing : M2  
Executing : M1  
Executing : C25  
Executing : C14  
Executing : M4  
Executing : M5
```

Conclusions

- Transformations between different formalisms encompass more than transforming syntax
- We exploited the semantic openness of UML state machines to ensure trace equivalence
- No formal proof is given, therefore it cannot be guaranteed that the semantic gap is completely bridged

Reference



M.F. van Amstel, M.G.J. van den Brand, Z. Protić, and T. Verhoeff.

Transforming Process Algebra Models into UML State Machines: Bridging a Semantic Gap?

To appear in: Proceedings of the International Conference on Model Transformations, 2008.