Formal Relationship between Petri Net and Graph Transformation Systems based on Functor between $\mathcal{M}$-adhesive Categories

Maria Maximova, Hartmut Ehrig, and Claudia Ermel

Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany
{mascham,ehrig,lieske}@cs.tu-berlin.de

Motivation

For typed attributed graph transformation systems the tool environment AGG [5] implements the execution and analysis of graph transformations [1]. A corresponding tool for Petri net transformation systems, called RON-Environment [4], simulates Petri net transformations based on corresponding graph transformations using AGG. Unfortunately, the correspondence between Petri net and graph transformations is handled at an informal level up to now. A first approach to relate Petri nets and graph transformation systems has been proposed by Kreowski in [2], where the Petri net firing behavior is expressed by graph transformation rules. We now want to consider Petri net transformations in addition. The purpose of this work is to establish a formal relationship between two corresponding $\mathcal{M}$-adhesive transformation systems enabling us in particular to translate Petri net transformations into graph transformations and, vice versa, to create Petri net transformations from graph transformations. This should work for different kinds of Petri nets. For this reason we propose to define suitable functors, called $\mathcal{M}$-functors, between different $\mathcal{M}$-adhesive categories and to investigate properties, which allow to translate and create transformations of the corresponding $\mathcal{M}$-adhesive transformation systems.

Translation and Creation of Transformations using $\mathcal{M}$-Functors

A weak adhesive HLR category $(\mathcal{C}, \mathcal{M})$ [1], short $\mathcal{M}$-adhesive category, consists of a category $\mathcal{C}$ together with a suitable class $\mathcal{M}$ of monos. In order to study translation and creation of transformations between different $\mathcal{M}$-adhesive transformation systems we introduce the notion of an $\mathcal{M}$-functor.

Definition 1 ($\mathcal{M}$-Functor).
A functor $\mathcal{F} : (\mathcal{C}_1, \mathcal{M}_1) \rightarrow (\mathcal{C}_2, \mathcal{M}_2)$ between $\mathcal{M}$-adhesive categories is called $\mathcal{M}$-functor, if we have $\mathcal{F}(\mathcal{M}_1) \subseteq \mathcal{M}_2$ and $\mathcal{F}$ preserves pushouts along $\mathcal{M}$-morphisms.

We want to translate transformations from $AS_1 = (\mathcal{C}_1, \mathcal{M}_1, P_1)$ to $AS_2 = (\mathcal{C}_2, \mathcal{M}_2, P_2)$ with $\mathcal{M}$-adhesive category $(\mathcal{C}_2, \mathcal{M}_2)$ and suitable productions P_2. This can be done using an $\mathcal{M}$-functor $\mathcal{F} : (\mathcal{C}_1, \mathcal{M}_1) \rightarrow (\mathcal{C}_2, \mathcal{M}_2)$ for $P_2 = \mathcal{F}(P_1)$.

Theorem 1 (Translation of Transformations).
An $\mathcal{M}$-functor $\mathcal{F} : (\mathcal{C}_1, \mathcal{M}_1) \to (\mathcal{C}_2, \mathcal{M}_2)$ translates productions, applicability of productions, construction of (direct) transformations, as well as parallel and sequential independence of transformations.

Vice versa, we have the following result for the creation of transformations.

Theorem 2 (Creation of Transformations).
Given an $\mathcal{M}$-functor $\mathcal{F} : (\mathcal{C}_1, \mathcal{M}_1) \to (\mathcal{C}_2, \mathcal{M}_2)$ with initial pushouts in $(\mathcal{C}_1, \mathcal{M}_1)$, which creates morphisms and preserves initial pushouts, then $\mathcal{F}$ creates applicability of productions, direct transformations, as well as parallel and sequential independence of transformations.

If we want to consider only (direct) transformations with injective matches, as in the case of Petri net transformations in the next section, then it is sufficient to define the functor $\mathcal{F}$ on injective morphisms only and to show modified conditions for the translation and creation of transformations.

Translation and Creation of Petri Net Transformations

According to our overall aim outlined in the last section, we want to construct a functor from Petri nets to typed attributed graphs, such that we are able to apply the results in Theorem 1 and Theorem 2.

For $(\mathcal{C}_1, \mathcal{M}_1)$ we take the category $(\mathbf{PTINet}, \mathcal{M}_1)$ of Petri nets with individual tokens and class $\mathcal{M}_1$ of injective morphisms, which is defined and shown to be $\mathcal{M}$-adhesive in [3]. Other choices for $(\mathcal{C}_1, \mathcal{M}_1)$ would be place/transition nets without initial marking or algebraic high-level nets [3]. For $(\mathcal{C}_2, \mathcal{M}_2)$ we take the category of typed attributed graphs $(\mathbf{AGraphs_{ATG}}, \mathcal{M}_2)$, which is shown to be $\mathcal{M}$-adhesive in [1] with specific attributed Petri net type graph $ATG = PNTG$.

An example for the application of a functor $\mathcal{F}$ on objects is shown in Figure 1, where NI is a Petri net with a and b being individual tokens on place p_1.

We can construct a functor $\mathcal{F} : \mathbf{PTINet}|_{\mathcal{M}_1} \to \mathbf{AGraphs_{PNTG}}|_{\mathcal{M}_2}$, but not an $\mathcal{M}$-functor $\mathcal{F} : (\mathbf{PTINet}, \mathcal{M}_1) \to (\mathbf{AGraphs_{PNTG}}, \mathcal{M}_2)$, because $\mathcal{F}$ is not well-defined on non-injective morphisms.

However, we are able to obtain translation and creation of Petri net transformations with injective matches in the sense of Theorem 1 and Theorem 2, because we can show the following non-trivial constructions and results:

1. Construction of functor $\mathcal{F} : \mathbf{PTINet}|_{\mathcal{M}_1} \to \mathbf{AGraphs_{PNTG}}|_{\mathcal{M}_2}$,
2. $\mathcal{F}$ translates pushouts of $\mathcal{M}_1$-morphisms in $(\mathbf{PTINet}, \mathcal{M}_1)$ into pushouts of $\mathcal{M}_2$-morphisms in $(\mathbf{AGraphs_{PNTG}}, \mathcal{M}_2)$,
3. $\mathcal{F}$ creates $\mathcal{M}_1$-morphisms,
4. $\mathcal{F}$ preserves initial pushouts over $\mathcal{M}_1$-morphisms.

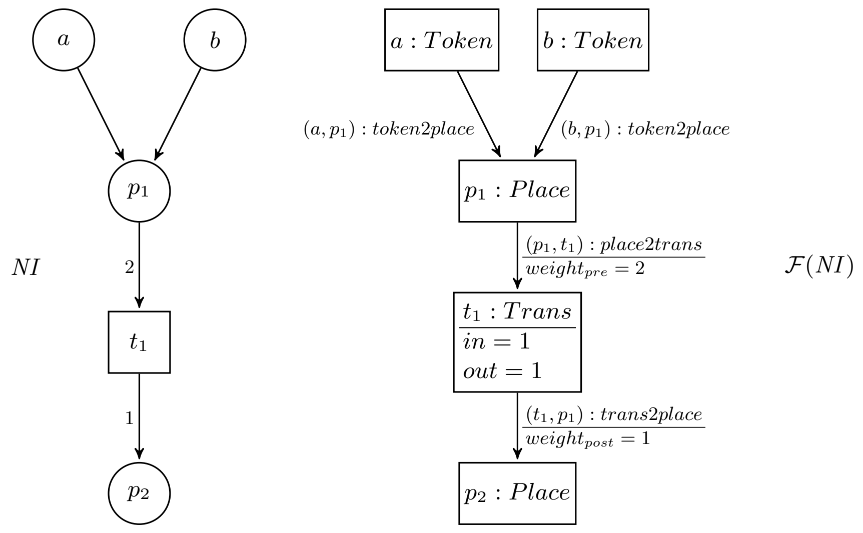

Fig. 1. PTI net NI and its translation into a typed attributed graph $\mathcal{F}(NI)$

Conclusion and Future Work

In addition to Theorem 1 and Theorem 2 above, we are able to show that the functor $\mathcal{F}$ translates and creates applicability of productions, direct transformations, as well as parallel and sequential independence.

In the long run, this should enable us to analyze interesting properties of Petri net transformation systems, like independence, termination and local confluence, using corresponding results and analysis tools like AGG for graph transformation systems. Moreover, it is interesting to study the relationship between other $\mathcal{M}$-adhesive transformation systems using this approach.

References

1. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in Theor. Comp. Science, Springer (2006), http://www.springer.com/3-540-31187-4
2. Kreowski, H.J.: A Comparison between Petri Nets and Graph Grammars. In: 5th International Workshop on Graph-Theoretic Concepts in Computer Science. LNCS, vol. 100, pp. 1–19. Springer (1981)
3. Modica, T., Gabriel, K., Ehrig, H., Hoffmann, K., Shareef, S., Ermel, C., Golas, U., Hermann, F., Biermann, E.: Low- and High-Level Petri Nets with Individual Tokens. Tech. Rep. 2009/13, Technische Universität Berlin (2009), http://www.eecs.tu-berlin.de/menue/forschung/forschungsberichte/2009
4. TFS-Group, TU Berlin: Reconfigurable Object Nets Environment (2007), http://www.tfs.cs.tu-berlin.de/roneditor
5. TFS-Group, TU Berlin: AGG (2009), http://tfs.cs.tu-berlin.de/agg