Proceedings of the 4th International Workshop on Petri Nets and Graph Transformation (PNGT 2010)

Formal Relationship between Petri Net and Graph Transformation Systems based on Functors between \( \mathcal{M} \)-adhesive Categories

Maria Maximova, Hartmut Ehrig and Claudia Ermel

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

Maria Maximova, Hartmut Ehrig and Claudia Ermel

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

Abstract: Various kinds of graph transformations and Petri net transformation systems are examples of $\mathcal{M}$-adhesive transformation systems based on $\mathcal{M}$-adhesive categories, generalizing weak adhesive HLR categories. For typed attributed graph transformation systems, the tool environment AGG allows the modeling, the simulation and the analysis of graph transformations. A corresponding tool for Petri net transformation systems, the RON-Environment, has recently been developed which implements and simulates Petri net transformations based on corresponding graph transformations using AGG. Up to now, the correspondence between Petri net and graph transformations is handled on an informal level. The purpose of this paper is to establish a formal relationship between the corresponding $\mathcal{M}$-adhesive transformation systems, which allow the translation of Petri net transformations into graph transformations with equivalent behavior, and, vice versa, the creation of Petri net transformations from graph transformations. Since this is supposed to work for different kinds of Petri nets, we propose to define suitable functors, called $\mathcal{M}$-functors, between different $\mathcal{M}$-adhesive categories and to investigate properties allowing us the translation and creation of transformations of the corresponding $\mathcal{M}$-adhesive transformation systems.

Keywords: $\mathcal{M}$-adhesive transformation system, equivalence, graph transformation, Petri net transformation, $\mathcal{M}$-adhesive category

1 Introduction

Modeling the adaptation of a dynamic system to a changing environment gets more and more important. Application areas cover e.g. computer supported cooperative work, multi agent systems or mobile networks. One approach to combine formal modeling of dynamic systems and controlled model adaption are reconfigurable Petri nets. The main idea is the stepwise development of place/transition nets by applying net transformation rules [EHP+08, PEHP08]. This approach increases the expressiveness of Petri nets and allows in addition to the well known token game a formal description and analysis of structural changes.

Rule-based Petri net transformation is related to graph transformation [EEPT06]. For typed attributed graph transformation systems, the well-established tool AGG [AGG09] allows the modeling, the simulation and the analysis of graph transformations. Recently, a tool for reconfigurable Petri nets, called RON-Tool [TFS07, BEHM07] (Reconfigurable Object Nets), exe-
Translation and Creation of Transformations

cutest and analyzes Petri net transformations based on corresponding graph transformations using AGG. As a matter of fact, the correspondence between Petri net and graph transformations is handled on an informal level up to now. Since both graph and net transformation systems are formally defined, the aim of this paper is to propose formal criteria ensuring a semantical correspondence of reconfigurable Petri nets and their corresponding representations as graph transformation systems.

An $\mathcal{M}$-adhesive transformation system is a general categorical transformation framework based on $\mathcal{M}$-adhesive categories, which rely on a class $\mathcal{M}$ of monomorphisms, generalizing weak adhesive HLR categories. The double-pushout approach, based on categorical constructions, is a suitable description of transformations leading to results like the Local Church-Rosser, Parallelism, Concurrency, Embedding, Extension, and Local Confluence Theorems [EEPT06].

A set of rules over an $\mathcal{M}$-adhesive category according to the double-pushout approach constitutes an $\mathcal{M}$-adhesive transformation system [EGH10].

Aiming for a more general approach to ensure a semantical correspondence of different transformation systems, we establish a formal relationship between two corresponding $\mathcal{M}$-adhesive transformation systems. This correspondence allows us especially the translation of Petri net transformations into graph transformations and, vice versa, the creation of Petri net transformations from graph transformations in order to analyze the behavior of Petri net transformation systems by analyzing their translation in terms of typed attributed graph transformation systems using the tool AGG [AGG09]. We propose to define suitable functors, called $\mathcal{M}$-functors, between different $\mathcal{M}$-adhesive categories and to investigate properties, which allow us the translation and creation of transformations of the corresponding $\mathcal{M}$-adhesive transformation systems.

The paper is structured as follows: Section 3 introduces the formal notions $\mathcal{M}$-adhesive transformation systems and $\mathcal{M}$-functors. The first main result given in Section 4 states that an $\mathcal{M}$-functor translates rules in a way that applicability and transformation results are translated as well. Vice versa, the second main result states that an $\mathcal{M}$-functor also creates applicability of rules in the other direction. Section 5 applies these new main result to the translation and creation of Petri net transformations by constructing and analyzing an $\mathcal{M}$-functor from the category of place/transition nets to the category of typed attributed graphs with corresponding type graph1. In Section 6, we conclude and propose interesting future research directions.

2 Related Work

In [MM90], Meseguer and Montanari represented Petri nets as graphs equipped with operations for composition of transitions. They introduced categories for Petri nets with and without initial markings and functors expressing duality and invariants. Their constructions provide a formal basis for expressing concurrency in terms of algebraic structures over graphs and categories. Based on categorical Petri nets, in [DS02], Petri nets are related to automata with concurrency relations by establishing a correspondence as coreflection between the associated categories. A first approach to relate Petri nets and graph transformation systems has been proposed by Kowalski in [Kre81], where Petri net firing behavior is expressed by graph transformation rules. In our approach, we want to consider Petri net transformations in addition. Moreover, we aim for a

1 For the results in Section 5, we give only proof ideas. More detailed proofs are given in [MEE11].
more general approach that establishes a semantical correspondence not only between Petri net and graph transformation systems but between any kind of formally defined rule-based transformation systems that can be generalized as \( \mathcal{M} \)-adhesive transformation systems.

In order to transform not only graphs, but also high-level structures as Petri nets and algebraic specifications, high-level replacement (HLR) categories were established in [EHKP91a, EHKP91b], which require a list of so-called HLR properties to hold. They were based on a morphism class \( \mathcal{M} \) used for the rule morphisms. This framework allowed a rich theory of transformations for all HLR categories, but the HLR properties were difficult and lengthy to verify for each category. Combining adhesive categories [LS04] and HLR categories lead to (weak) adhesive HLR categories in [EHPP06] and to \( \mathcal{M} \)-adhesive categories in [EGH10], where a subclass \( \mathcal{M} \) of monomorphisms is considered and only pushouts over \( \mathcal{M} \)-morphisms have to fulfill the \textit{van Kampen property} (a certain compatibility of pushouts and pullbacks). Not only many kinds of graphs, but also different kinds of place/transition nets and algebraic high-level nets are \( \mathcal{M} \)-adhesive and also weak adhesive HLR categories which allows the application of the theory to all these kinds of structures [EEPT06, PE07, MGE+09]. In fact, all results in [EEPT06] for weak adhesive HLR categories are also valid for \( \mathcal{M} \)-adhesive categories [EGH10].

\section{\( \mathcal{M} \)-Adhesive Categories, Transformation Systems and \( \mathcal{M} \)-Functors}

An \( \mathcal{M} \)-adhesive category [EGH10], consists of a category \( \mathcal{C} \) together with a class \( \mathcal{M} \) of monomorphisms as defined in Definition 1 below. The concept of \( \mathcal{M} \)-adhesive categories generalises that of weak adhesive, adhesive HLR and adhesive categories [LS04]. The category of typed attributes graphs and several categories of Petri nets are weak adhesive HLR (see [EEPT06]) and hence also \( \mathcal{M} \)-adhesive.

\begin{definition}[\( \mathcal{M} \)-Adhesive Category]
An \( \mathcal{M} \)-adhesive category \( (\mathcal{C}, \mathcal{M}) \) is a category \( \mathcal{C} \) together with a class \( \mathcal{M} \) of monomorphisms satisfying

\begin{itemize}
  \item \( \mathcal{C} \) has pushouts (POs) and pullbacks (PBs) along \( \mathcal{M} \)-morphisms,
  \item \( \mathcal{M} \) is closed under composition, decomposition, POs and PBs,
  \item POs along \( \mathcal{M} \)-morphisms are \( \mathcal{M} \)-VK-squares, i.e. the VK-property holds for all commutative cubes, where the given PO with \( m \in \mathcal{M} \) is in the bottom, the back faces are PBs and all vertical morphisms \( a, b, c \) and \( d \) are in \( \mathcal{M} \).
\end{itemize}

\end{definition}

\begin{figure}
\centering
\begin{tikzpicture}
  \node (A) at (0,0) {A};
  \node (B) at (1,1) {B};
  \node (C) at (-1,1) {C};
  \node (D) at (-1,-1) {D};

  \draw[->] (A) -- node[above] {$a$} (B);
  \draw[->] (A) -- node[left] {$g$} (D);
  \draw[->] (B) -- node[right] {$b$} (D);

  \draw[->] (C) -- node[below] {$f$} (A);
  \draw[->] (C) -- node[right] {$d$} (D);

  \draw[->] (D) -- node[left] {$c$} (C);

  \draw[->] (A) -- node[above] {$m$} (B);
  \draw[->] (C) -- node[above] {$n$} (D);

  \draw[->] (A) -- node[below] {$m'$} (B);
  \draw[->] (C) -- node[below] {$n'$} (D);

  \draw[->] (A) -- node[left] {$f'$} (C);
  \draw[->] (A) -- node[right] {$g'$} (D);

  \draw[->] (B) -- node[below] {$m$} (D);

\end{tikzpicture}
\caption{Example of a \( \mathcal{M} \)-VK-square}
\end{figure}

\begin{definition}[\( \mathcal{M} \)-Adhesive Transformation System and Independence]
Given an \( \mathcal{M} \)-adhesive category \( (\mathcal{C}, \mathcal{M}) \).

\begin{itemize}
  \item An \( \mathcal{M} \)-adhesive transformation system \( AS = (\mathcal{C}, \mathcal{M}, P) \) has in addition a set \( P \) of productions of the form \( \rho = (L \xleftarrow{l} K \xrightarrow{r} R) \) with \( l, r \in \mathcal{M} \).
\end{itemize}

\end{definition}
A direct transformation $G \xrightarrow{m} H$ via production $\rho$ and match $m$ consists of two POs (1) and (2) as shown in the diagram to the right, where $n : R \rightarrow H$ is called comatch of $m$. A production $\rho$ is applicable via $m$ to $G$, if we have a PO complement $D$ in (1), such that (1) becomes a PO.

- Two (direct) transformations $G \xrightarrow{\rho_1,m_1} H_1$ and $G \xrightarrow{\rho_2,m_2} H_2$ are called parallel independent, if there are morphisms $d_{12} : L_1 \rightarrow D_2$, $d_{21} : L_2 \rightarrow D_1$ such that $l_1^* \circ d_{21} = m_2$ and $l_2^* \circ d_{12} = m_1$. Dually $G \xrightarrow{\rho_1,m_1} H_1$ and $H_1 \xrightarrow{\rho_2,m_2} H_2$ are sequentially independent if $H_1 \xrightarrow{\rho_1^{-1}, \rho_1} G$ and $H_1 \xrightarrow{\rho_2, \rho_2^{-1}} H_2$ are parallel independent, where $\rho_1^{-1} = (R_1 \xleftarrow{l_1} K_1 \rightarrow L_1)$ and $n_1$ is the comatch of $m_1$.

$$
\begin{array}{c}
\begin{array}{c}
R_1 \\
\downarrow \\
H_1 \\
D_1
\end{array}
\quad
\begin{array}{c}
K_1 \\
\downarrow \\
G \\
L_1
\end{array}
\quad
\begin{array}{c}
L_2 \\
\downarrow \\
K_2 \\
D_2
\end{array}
\quad
\begin{array}{c}
R_2 \\
\downarrow \\
H_2
\end{array}

\end{array}
$$

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. An $\mathcal{M}$-functor establishes a semantical correspondence between different $\mathcal{M}$-adhesive transformation systems.

**Definition 3 ($\mathcal{M}$-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 $\mathcal{F}(\mathcal{M}_1) \subseteq \mathcal{M}_2$ and $\mathcal{F}$ preserves pushouts along $\mathcal{M}$-morphisms.

On purpose we don’t require that an $\mathcal{M}$-functor preserves pullbacks along $\mathcal{M}$-morphisms, VK-squares, or other properties, but later additional properties of $\mathcal{F}$ will be required in order to achieve specific results.

**Remark 1**
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. Moreover, this restriction is necessary, if $\mathcal{F}$ is not well-defined for non-injective morphisms.

For this case we need to define a special kind of an $\mathcal{M}$-functor: a restricted $\mathcal{M}$-functor.

**Definition 4 (Restricted $\mathcal{M}$-Functor)**
A functor $\mathcal{F} : (\mathcal{C}_i, \mathcal{M}_i) \rightarrow (\mathcal{C}_2, \mathcal{M}_2)$ between $\mathcal{M}$-adhesive categories $(\mathcal{C}_1, \mathcal{M}_1)$ and $(\mathcal{C}_2, \mathcal{M}_2)$ with $\mathcal{C}_i$, the restriction of $\mathcal{C}_i$ to $\mathcal{M}_i$-morphisms for $i = 1, 2$ is called a restricted $\mathcal{M}$-functor, if $\mathcal{F}(\mathcal{M}_1) \subseteq \mathcal{M}_2$ and $\mathcal{F}$ translates POs along $\mathcal{M}_1$-morphisms in $(\mathcal{C}_1, \mathcal{M}_1)$ into POs along $\mathcal{M}_2$-morphisms in $(\mathcal{C}_2, \mathcal{M}_2)$.
4 Translation and Creation of Transformations

To obtain a semantical correspondence between any two transformation systems we need to ensure that the respective transformation systems together with their relevant properties are translated and reflected properly.

Given an $\mathcal{M}$-adhesive transformation system $\mathcal{A}S_1 = (C_1, \mathcal{M}_1, P_1)$ with an $\mathcal{M}$-adhesive category $(C_1, \mathcal{M}_1)$ and productions $P_1$. We want to translate transformations from $\mathcal{A}S_1$ to $\mathcal{A}S_2 = (C_2, \mathcal{M}_2, P_2)$ with $\mathcal{M}$-adhesive category $(C_2, \mathcal{M}_2)$ and suitable productions $P_2$. This can be done using an $\mathcal{M}$-functor $\mathcal{F}: (C_1, \mathcal{M}_1) \rightarrow (C_2, \mathcal{M}_2)$ for $P_2 = \mathcal{F}(P_1)$.

**Theorem 1** (Translation of Transformations)

An $\mathcal{M}$-functor $\mathcal{F}: (C_1, \mathcal{M}_1) \rightarrow (C_2, \mathcal{M}_2)$ translates applicability of productions, construction of (direct) transformations, as well as parallel and sequential independence of transformations.

**Proof.**

$\mathcal{A}S_2 = (C_2, \mathcal{M}_2, \mathcal{F}(P_1))$ is a well-defined $\mathcal{M}$-adhesive transformation system, because $\mathcal{F}$ translates $\mathcal{M}_1$-morphisms into $\mathcal{M}_2$-morphisms for the productions and each direct transformation $G \xrightarrow{\rho,m} H$ in $\mathcal{A}S_1$ given by pushouts (1) and (2) leads to a direct transformation $\mathcal{F}(G) \xrightarrow{\mathcal{F}(\rho),\mathcal{F}(m)} \mathcal{F}(H)$ in $\mathcal{A}S_2$ given by pushouts (3) and (4), because $\mathcal{F}$ preserves pushouts along $\mathcal{M}$-morphisms.

$$
\begin{array}{ccc}
L & \xleftarrow{l} & K \xrightarrow{r} R \\
\downarrow{m} & (1) & (2) & \Downarrow{\mathcal{F}} \\
G & \xleftarrow{D} & H
\end{array}

\Rightarrow

\begin{array}{ccc}
\mathcal{F}(L) & \xleftarrow{\mathcal{F}(l)} & \mathcal{F}(K) \xrightarrow{\mathcal{F}(r)} \mathcal{F}(R) \\
\downarrow & (3) & (4) & \downarrow \\
\mathcal{F}(G) & \xleftarrow{\mathcal{F}(D)} & \mathcal{F}(H)
\end{array}
$$

Moreover, the functor property of $\mathcal{F}$ implies that $\mathcal{F}$ translates parallel and sequential independence of transformations.

As shown above, we need for translation of transformations from $\mathcal{A}S_1$ to $\mathcal{A}S_2$ only the basic properties of an $\mathcal{M}$-functor. This is no longer true for creation of transformations in $\mathcal{A}S_1$ from transformations in $\mathcal{A}S_2$ with $P_2 = \mathcal{F}(P_1)$ as above.

**Definition 5** (Creation of Applicability and Direct Transformations)

1. An $\mathcal{M}$-functor $\mathcal{F}: (C_1, \mathcal{M}_1) \rightarrow (C_2, \mathcal{M}_2)$ creates applicability of a production $\rho = (L \xleftarrow{L} K \xrightarrow{r} R)$ to object $G$, if applicability of $\mathcal{F}(\rho)$ to $\mathcal{F}(G)$ with match $m': \mathcal{F}(L) \rightarrow \mathcal{F}(G)$ implies applicability of $\rho$ to $G$ with some match $m: L \rightarrow G$ and $\mathcal{F}(m) = m'$.

2. $\mathcal{F}$ creates direct transformations, if for each direct transformation $\mathcal{F}(G) \xrightarrow{\mathcal{F}(\rho),\mathcal{F}(m)} H'$ in $\mathcal{A}S_2$ there is a direct transformation $G \xrightarrow{\rho,m} H$ in $\mathcal{A}S_1$ with $\mathcal{F}(m) = m'$ and $\mathcal{F}(H) \cong H'$ leading to $\mathcal{F}(G) \xrightarrow{\mathcal{F}(\rho),\mathcal{F}(m)} \mathcal{F}(H)$ in $\mathcal{A}S_2$: 
Let \( \mathcal{F} \) be a \( \mathcal{M} \)-functor. Then, if the following “gluing condition” is satisfied:

\[
\mathcal{F}(L) \xleftarrow{f(1)} \mathcal{F}(K) \xrightarrow{f(2)} \mathcal{F}(R)
\]

\[
m' \downarrow \quad (1) \quad \downarrow \quad (2) \quad \Rightarrow \quad m \downarrow \quad (3) \quad \downarrow \quad (4)
\]

\[
\mathcal{F}(G) \xleftarrow{h} G \quad \cdots \quad D \quad \cdots \quad H
\]

3. \( \mathcal{F} \) creates parallel (and similarly sequential) independence, if parallel independence of

\[
\mathcal{F}(H_1) \xleftarrow{\mathcal{F}(p_1), \mathcal{F}(m_1)} \mathcal{F}(G) \xrightarrow{\mathcal{F}(p_2), \mathcal{F}(m_2)} \mathcal{F}(H_2)
\]

in \( \mathcal{A} \mathcal{S}_2 \) implies parallel independence of \( H_1 \xleftarrow{p_1,m_1} G \xrightarrow{p_2,m_2} H_2 \) in \( \mathcal{A} \mathcal{S}_1 \).

**Remark 2**

If \( \mathcal{F} \) creates parallel (sequential) independence, then \( \mathcal{F} \) characterises parallel (sequential) independence, i.e., parallel (sequential) independence in \( \mathcal{A} \mathcal{S}_1 \) is equivalent to parallel (sequential) independence in \( \mathcal{A} \mathcal{S}_2 \), because \( \mathcal{F} \) already preserves parallel (sequential) independence by Theorem 1.

In the following we formulate the properties for an \( \mathcal{M} \)-functor \( \mathcal{F} \), such that we have creation of applicability, direct transformations and parallel (sequential) independence. But first we review the notion of initial pushouts motivated by Remark 3 below.

**Definition 6** (Initial Pushout)

Given a morphism \( f : G \to G' \) in an \( \mathcal{M} \)-adhesive category \( (\mathcal{C}, \mathcal{M}) \). (1) is an initial pushout (IPO) over \( f \) with boundary \( B \), context \( C \) and \( b, c \in \mathcal{M} \), if

\[
\exists ! b^* : B \to B^* \quad \text{such that we have creation of applicability, direct transformations and parallel (sequential) independence. But first we}
\]

review the notion of initial pushouts motivated by Remark 3 below.

**Definition 7** (Properties of \( \mathcal{M} \)-Functors)

1. An \( \mathcal{M} \)-functor \( \mathcal{F} : (\mathcal{C}_1, \mathcal{M}_1) \to (\mathcal{C}_2, \mathcal{M}_2) \) creates morphisms, if for all \( m' : \mathcal{F}(L) \to \mathcal{F}(G) \)

2. \( \mathcal{F} \) preserves initial pushouts, if for each initial pushout \( \text{IPO} \) (1) over \( m : L \to G \), also (2) is initial pushout over \( \mathcal{F}(m) : \mathcal{F}(L) \to \mathcal{F}(G) \).
This leads to the following theorem on creation of transformations by $\mathcal{M}$-functors:

**Theorem 2** (Creation of Transformations)

Given an $\mathcal{M}$-functor $\mathcal{F} : (\mathcal{C}_1, \mathcal{M}_1) \rightarrow (\mathcal{C}_2, \mathcal{M}_2)$ 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.

**Proof.**

1. $\mathcal{F}$ creates applicability of productions

Given $\rho = (L \xleftarrow{l} K \xrightarrow{r} R)$ and match $m' : \mathcal{F}(L) \rightarrow \mathcal{F}(G)$, s.t., $\mathcal{F}(\rho)$ is applicable to $m'$. Since $\mathcal{F}$ creates morphisms we have a unique $m : L \rightarrow G$ with $\mathcal{F}(m) = m'$. Let (1) be an initial pushout over $m$ in the diagram below. By assumption on $\mathcal{F}$, (2) is initial pushout over $\mathcal{F}(m)$ and (4), (5) are POs. This means, that $\mathcal{F}(\rho)$ is applicable to $m = \mathcal{F}(m)$. According to Remark 3, this implies the existence of $b'' : \mathcal{F}(B) \rightarrow \mathcal{F}(K)$ in $\mathcal{M}_2$ with $\mathcal{F}(l) \circ b'' = \mathcal{F}(b)$.

Since $\mathcal{F}$ creates morphisms there is a unique morphism $b' : B \rightarrow K$ with $\mathcal{F}(b') = b''$. Moreover, uniqueness of creation of morphisms implies $l \circ b' = b$ and hence $b' \in \mathcal{M}_1$ by decomposition property of $\mathcal{M}_1$. Hence the gluing condition is satisfied and we have applicability of $\rho$ to $G$ with match $m : L \rightarrow G$ and $\mathcal{F}(m) = m'$ with pushout complement $D$ in (3).

2. $\mathcal{F}$ creates direct transformations

Given the direct transformation $\mathcal{F}(G) \xrightarrow{\mathcal{F}(\rho), m'} H'$ in $\mathcal{A}_2$ by pushouts (4) and (5) in $(\mathcal{C}_2, \mathcal{M}_2)$. We have already constructed pushout (3) in $(\mathcal{C}_1, \mathcal{M}_1)$ and can construct pushout (6) along $r \in \mathcal{M}_1$ leading to a direct transformation $G \xrightarrow{\rho, m} H$. Since $\mathcal{F}$ preserves pushouts along $\mathcal{M}$-morphisms and pushout complements in $(\mathcal{C}_2, \mathcal{M}_2)$ and they are unique up to isomorphism. We have $\mathcal{F}(D) \cong D'$, $\mathcal{F}(H) \cong H'$ and hence also $\mathcal{F}(G) \xrightarrow{\mathcal{F}(\rho), \mathcal{F}(m)} \mathcal{F}(H)$ in $\mathcal{A}_2$.

3. $\mathcal{F}$ creates parallel (sequential) independence

By parallel independence of $\mathcal{F}(H_1) \xleftarrow{\mathcal{F}(\rho_1), \mathcal{F}(m_1)} \mathcal{F}(G) \xrightarrow{\mathcal{F}(\rho_2), \mathcal{F}(m_2)} \mathcal{F}(H_2)$ in $\mathcal{A}_2$ we have mor-
phisms $d'_{12} : \mathcal{F}(L_1) \rightarrow \mathcal{F}(D_2)$ with $\mathcal{F}(l'_2) \circ d'_{12} = \mathcal{F}(m_1)$ and $d'_{21} : \mathcal{F}(L_2) \rightarrow \mathcal{F}(D_1)$ with $\mathcal{F}(l'_1) \circ d'_{21} = \mathcal{F}(m_2)$ leading to corresponding morphisms $d_{12} : L_1 \rightarrow D_2$ and $d_{21} : L_2 \rightarrow D_1$ with $l'_2 \circ d_{12} = m_1$ and $l'_1 \circ d_{21} = m_2$, because $\mathcal{F}$ creates morphisms uniquely and preserves composition.

**Remark 4** For the case described in the Remark 1 we have to show for Theorem 1 that $\mathcal{F}$ translates pushouts of $\mathcal{M}_1$-morphisms in $(C_1, \mathcal{M}_1)$ into pushouts of $\mathcal{M}_2$-morphisms in $(C_2, \mathcal{M}_2)$. For Theorem 2 we need in addition, that $\mathcal{F}$ creates $\mathcal{M}$-morphisms, i.e., for each $(m' : \mathcal{F}(L) \rightarrow \mathcal{F}(G)) \in \mathcal{M}_2$ there is exactly one morphism $(m : L \rightarrow G) \in \mathcal{M}_1$ with $\mathcal{F}(m) = m'$ and $\mathcal{F}$ preserves initial pushouts over $\mathcal{M}_1$-morphisms. Note, that we cannot replace the $\mathcal{M}$-adhesive categories $(C_i, \mathcal{M}_i)$ for $i = 1, 2$ by $(C_i, \mathcal{M}_i, \mathcal{M}_i)$, because $(C_i, \mathcal{M}_i, \mathcal{M}_i)$ are in general not $\mathcal{M}$-adhesive.

## 5 Translation and Creation of Petri Net Transformations

According to our overall aim in Section 1 we want to construct a functor from Petri nets to typed attributed graphs and show how to apply the main results of Theorem 1 and Theorem 2 in order to translate and create Petri net transformations using graph transformations. For this purpose we review on one hand the $\mathcal{M}$-adhesive categories $(\text{PTINet}, \mathcal{M}_1)$ of Petri nets with individual tokens and class $\mathcal{M}_1$ of all injective morphisms, which is defined and shown to be $\mathcal{M}$-adhesive in [MGE+09]. On the other hand we review typed attributed graphs $(\text{AGraphs}_{ATG}, \mathcal{M}_2)$, which are shown to be $\mathcal{M}$-adhesive in [EEPT06] and we define a suitable attributed Petri net type graph $ATG = PNTG$. Moreover we construct a functor $\mathcal{F}$ between both categories, which, however, is only defined on injective morphisms $\mathcal{M}_1$.

Note, that we do not use Petri nets with “classical initial markings”, known as Petri net systems [Rei85], because the corresponding $\mathcal{M}$-adhesive category requires a class $\mathcal{M}$ leading to Petri net rules which are marking preserving. Marking preserving rules are not adequate to model firing steps as direct transformations since tokens must not be created or deleted. Other choices for $(C_1, \mathcal{M}_1)$ would be Petri nets without initial marking or algebraic high-level nets (see [EEPT06, Rei85, MGE+09]).

In fact, we can construct a functor $\mathcal{F} : \text{PTINet} \mid \mathcal{M}_1 \rightarrow \text{AGraphs}_{PNTG} \mid \mathcal{M}_2$, between the categories restricted to $\mathcal{M}$-morphisms, but not an $\mathcal{M}$-functor $\mathcal{F} : (\text{PTINet}, \mathcal{M}_1) \rightarrow (\text{AGraphs}_{PNTG}, \mathcal{M}_2)$, because $\mathcal{F}$ is not well-defined on non-injective morphisms (see counterexample in Figure 1 below, where $\mathcal{F}(f)$ does not preserve attributes in and $w_{\text{pre}}$). This means, we proceed as discussed in Remark 4, which allows the application of Theorem 1 and Theorem 2 in order to obtain translation and creation of Petri net transformations with injective morphisms. For application of Theorem 1 we need steps 1.-5., and for Theorem 2 in addition steps 6. and 7.

1. Definition of Petri nets with individual Tokens: $\text{PTINet}$.
2. Definition of typed attributed graphs over Petri net type graph $PNTG$: $\text{AGraphs}_{PNTG}$.
3. Translation of PTI nets into $PNTG$-typed attributed graphs (definition of functor $\mathcal{F}$ on objects).
4. Translation of restricted $\text{PTINet}$-morphisms into restricted $\text{AGraphs}_{PNTG}$-morphisms (definition of functor $\mathcal{F} : \text{PTINet} \mid \mathcal{M}_1 \rightarrow \text{AGraphs}_{PNTG} \mid \mathcal{M}_2$ on morphisms).
5. \( \mathcal{F} \) translates pushouts of \( \mathcal{M}_1 \)-morphisms in \((\text{PTINet}, \mathcal{M}_1)\) into pushouts of \( \mathcal{M}_2 \)-morphisms in \((\text{AGraph}_{\text{PNTG}}, \mathcal{M}_2)\).

6. \( \mathcal{F} \) creates \( \mathcal{M}_1 \)-morphisms.

7. \( \mathcal{F} \) preserves initial pushouts over \( \mathcal{M}_1 \)-morphisms.

Figure 1: Counterexample for general (non-injective) morphisms

5.1 Petri Nets with Individual Tokens: PTINet

For classical place / transition (P/T) nets \( N \) we adopt the approach of Meseguer and Montanari [MM90] using free commutative monoids \( P^\oplus \) over \( P \), where \( N = (P, T, \text{pre}, \text{post}) \) with places \( P \), transitions \( T \), functions \( \text{pre}, \text{post} : T \to P^\oplus \) and markings \( M \in P^\oplus \).

Petri nets \( NI = (P, T, \text{pre}, \text{post}, I, m) \) with individual tokens are place/transition nets \( N = (P, T, \text{pre}, \text{post}) \) together with a set \( I \) of individual tokens and a marking function \( m : I \to P \) assigning a place \( m(x) \in P \) to each \( x \in I \). Therefore two (or more) different individual tokens \( x, y \in I \) may be on the same place, i.e. \( m(x) = m(y) \), while in the standard “collective token approach” the marking \( M \in P^\oplus \) tells us only how many tokens we have on each place, but we are not able to distinguish between two tokens on the same place.

A formal definition of a Petri net with individual tokens is as follows ([MGE+09]).

Definition 8 (Petri Net with Individual Tokens)

A Petri net with individual tokens \( NI = (P, T, \text{pre}, \text{post}, I, m) \) is given by a classical P/T net \( N = (P, T, \text{pre} : T \to P^\oplus, \text{post} : T \to P^\oplus) \), where \( P^\oplus \) is the free commutative monoid over \( P \), a (possibly infinite) set of individual tokens \( I \), and the marking function \( m : I \to P \), assigning to each individual token \( x \in I \) the corresponding place \( m(x) \in P \).

PTINet-morphisms now define not only a mapping between two P/T nets but also between their individual tokens:

Definition 9 (PTINet-Morphism)

A PTINet-morphism \( f : NI_1 \to NI_2 \) is given by a triple of functions \( f = (f_P : P_1 \to P_2, f_T : T_1 \to T_2, f_I : I_1 \to I_2) \), such that the following diagrams commute with \( \text{pre} \) and \( \text{post} \) respectively.
Translation and Creation of Transformations

\[ T_1 \xrightarrow{f_{\text{fr}}} P_1 \oplus T_2 \xrightarrow{f_{\text{fr}}} P_2 \]

\[ I_1 \xrightarrow{f_1} P_1 \oplus I_2 \xrightarrow{f_2} P_2 \]

It is also shown in [MGE+09], that \((\text{PTINet}, \mathcal{M}_1)\) with the class \(\mathcal{M}_1\) of all injective morphisms is an \(\mathcal{M}\)-adhesive category, where pushouts and pullbacks are constructed componentwise (see Figure 2, where (1) is an example for a pushout in \((\text{PTINet}, \mathcal{M}_1)\), with individual tokens colored in black).

![Figure 2: PO in PTINet](image)

In the following we only consider the restriction of \(\text{PTINet}\) to \(\mathcal{M}_1\)-morphisms, \(\text{PTINet}|_{\mathcal{M}_1}\), in order to define the functor \(\mathcal{F}\) in Subsection 5.4, because \(\mathcal{F}\) is not well-defined on general morphisms. But we use the \(\mathcal{M}\)-adhesive category \((\text{PTINet}, \mathcal{M}_1)\) in order to define pushouts, because \((\text{PTINet}|_{\mathcal{M}_1}, \mathcal{M}_1)\) is not \(\mathcal{M}_1\)-adhesive due to the well-known fact, that the induced morphism of \(\mathcal{M}_1\)-morphisms is in general not an \(\mathcal{M}_1\)-morphism.

### 5.2 Typed Attributed Graphs over Petri Net Type Graph PNTG

According to [EEPT06] the category \((\text{AGraphs}_{\text{ATG}}: \mathcal{M}_2)\) of typed attributed graphs with class \(\mathcal{M}_2\) of all injective morphisms with isomorphism on the data type part is \(\mathcal{M}\)-adhesive, where pushouts along \(\mathcal{M}_2\)-morphisms are constructed componentwise in the graph part.
Objects in $\text{AGraphs}$ are pairs $(G,D)$ of an E-Graph $G$ with signature $E$ (shown to the right), and $\Sigma - \text{nat}$ data type $D$, where in the following we only use $D = T_\Sigma - \text{nat} \cong \text{NAT}$. This means, $G$ is given by $G = (V^G, V^D = \mathbb{N}, E^G, E^\text{NA}, E^\text{EA}, (s^G_j, t^G_j)_{j \in \{G,\text{NA},\text{EA}\}})$, where $V^G$ resp. $V^D$ are the graph resp. data nodes of $G$, $E^G, E^\text{NA}, E^\text{EA}$, resp. $s^G_j, t^G_j$ are the graph edges resp. node attribute and edge attribute edges of $G$ and $s^G_j, t^G_j$ are corresponding source and target functions for the edges.

In our case, the type graph $\text{ATG}$ is the Petri net type graph $\text{PNTG}$ shown in Figure 3 with data type signature $\Sigma - \text{nat}$ and algebra $\text{NAT}$ for rules and graphs, where the E-Graph of $\text{PNTG}$ is shown on the left and its attribute notation on the right of Figure 3. Objects in $\text{AGraphs}_{\text{PNTG}}$ are pairs $(AN, \text{type})$ with attributed graph $AN = (G,D)$ with $D = \text{NAT}$ and $\text{AGraphs}_{\text{PNTG}}$-morphism type $: (G,D) \rightarrow (\text{PNTG}, D_{\text{fin}})$ with final $\Sigma - \text{nat}$ data type $D_{\text{fin}}$. Morphisms in $\text{AGraphs}_{\text{PNTG}}$ are defined componentwise and are type compatible with morphisms in $\text{AGraphs}$. Four sample morphisms in $\text{AGraphs}_{\text{PNTG}}$ are shown in Figure 4, where a pushout is constructed.

### 5.3 Translation of PTI Nets into PNTG-typed Attributed Graphs

A formal definition of the functor $\mathcal{F}$ on objects is given as follows.

**Definition 10** (Translation of PTINet-Objects)
Given a PTI net $NI = (P, T, \text{pre}, \text{post}, I, m)$. We define the object $\mathcal{F}(NI) = ((G,\text{NAT}), \text{type})$ in $\text{AGraphs}_{\text{PNTG}}$ with type $: (G,\text{NAT}) \rightarrow (\text{PNTG}, D_{\text{fin}})$ and $G = (V^G, V^D = \mathbb{N}, E^G, E^\text{NA}, E^\text{EA}, (s^G_j, t^G_j)_{j \in \{G,\text{NA},\text{EA}\}})$ as follows, where we use the following abbreviations: $\text{token2place} \triangleq \text{to2p}$, $\text{place2trans} \triangleq \text{p2t}$, $\text{trans2place} \triangleq \text{t2p}$, $\text{weight}_{\text{pre}} \triangleq w_{\text{pre}}$, $\text{weight}_{\text{post}} \triangleq w_{\text{post}}$ and $\text{pre}(t)(p) = n_p \in \mathbb{N}$ for $\text{pre}(t) = \sum_{p \in P} n_p \cdot p \in \mathbb{P}$ and similar for $\text{post}(t)(p)$.
$V^G_P = P \cup T \cup I$

$E^G_P = E^G_{in2p} \cup E^G_{p2t} \cup E^G_{p2p}$

$E^G_{in2p} = \{(x,p) \in I \times P \mid m(x) = p\}$, $E^G_{p2t} = \{(p,t) \in P \times T \mid \text{pre}(t)(p) > 0\}$ and $E^G_{p2p} = \{(t,p) \in T \times P \mid \text{post}(t)(p) > 0\}$

$E^G_NA = E^G_{in} \cup E^G_{out}$

$E^G_{in} = \{(t,n,in) \mid (t,n) \in T \times N \land \bullet t = n\}$,

$E^G_{out} = \{(t,n,out) \mid (t,n) \in T \times N \land \bullet t = n\}$,

where $\bullet t$ and $\bullet t$ are the pre- and post-domains of $t \in T$ with cardinalities $|\bullet t|$ and $|\bullet t|$.

$E^G_{EA} = E^G_{w_{\text{pre}}} \cup E^G_{w_{\text{post}}}$

$E^G_{w_{\text{pre}}} = \{(p,t,n) \in E^G_P \times N \mid \text{pre}(t)(p) = n\}$

$E^G_{w_{\text{post}}} = \{(t,p,n) \in E^G_T \times N \mid \text{post}(t)(p) = n\}$

$s^G_G, t^G_G : E^G_P \rightarrow V^G_P$ defined by $s^G_G(a,b) = a$ resp. $t^G_G(a,b) = b$

$s^G_NA : E^G_NA \rightarrow V^G_P$, $t^G_NA : E^G_NA \rightarrow N$ defined by $s^G_NA(t,n,x) = t$ resp. $t^G_NA(t,n,x) = n$

$s^G_EA : E^G_EA \rightarrow E^G_P$ defined by $s^G_EA(p,t,n) = (p,t)$ and $s^G_EA(t,p,n) = (t,p)$

$t^G_EA : E^G_EA \rightarrow N$ defined by $t^G_EA(p,t,n) = n$ and $t^G_EA(t,p,n) = n$
The corresponding type-morphism is given in Definition 11 below.

An example for using the functor $\mathcal{F}$ on objects is shown in Figure 4, where the four typed attributed graphs are translations of the corresponding four PTI nets in Figure 2.

**Definition 11 (AGraphs$\text{PNTG}$-Morphism type)**

The AGraphs$\text{PNTG}$-morphism $\text{type} : (G,NAT) \rightarrow (\text{PNTG},D_{in})$ is given by final morphism of data types and $\text{type}^G : G \rightarrow \text{PNTG}$ given by E-graph morphism $\text{type}^G = (\text{type}_V^G,\text{type}_P^G,\text{type}_E^G,\text{type}_{EA^*},\text{type}_{EA^*})$ where

\[
\begin{align*}
\text{type}_V^G &: V^G \to V^{\text{PNTG}} \\
& \text{with } x \mapsto \text{Place} (x \in P), \quad x \mapsto \text{Trans} (x \in T), \quad x \mapsto \text{Token} (x \in I) \\
\text{type}_P^G &: N \to D_{in} \text{nat} \quad \text{with } x \mapsto \text{nat} (x \in N) \\
\text{type}_E^G &: E^G \to E^{\text{PNTG}} \quad \text{with } x \mapsto y \text{ for } x \in E^G_x \quad \text{and } y \in \{\text{to}2p, p2t, t2p\} \\
\text{type}_{EA^*} &: E_{NA}^* \to E_{\text{PNTG}} \quad \text{with } x \mapsto y \text{ for } x \in E^G_y \quad \text{and } y \in \{\text{w}_{pre}, \text{w}_{post}\}
\end{align*}
\]

**5.4 Translation of Restricted PTINet-Morphisms into Restricted AGraphs$\text{PNTG}$-Morphisms**

We now define the functor $\mathcal{F} : \text{PTINet}_{\mathcal{A}_1} \to \text{AGraphs}_{\text{PNTG}_{\mathcal{A}_1}}$ on injective morphisms. A counterexample for the translation of non-injective morphisms is given in Figure 1, examples for injective morphisms in Figure 2 and corresponding translated morphisms in Figure 4.

**Definition 12 (Translation of PTINet-Morphisms)**

For each PTINet-morphism $f : N_{I_1} \to N_{I_2}$ with $f = (f_P, f_T, f_I) \in \mathcal{A}_1$, i.e. $f_P, f_T, f_I$ injective, we define $\mathcal{F}(f) : \mathcal{F}(N_{I_1}) \to \mathcal{F}(N_{I_2})$ where $\mathcal{F}(N_{I_i}) = (V_{IG}, N, E_{IG}, E_{INA}, E_{IAE}, (s_{i, f}, t_{i, f}) \in \{G, N, E\})$ with $i = 1, 2$

by $\mathcal{F}(f) = f' = (f'_V, f'_P, f'_T, f'_{IA}, f'_{EA})$ with

\[
\begin{align*}
f'_V &: V_{IG} \to V_{I_2} \quad \text{with } V_{IG} = P_{i} \cup T_{i} \cup I_{i} \quad \text{for } i = 1, 2 \quad \text{by } f'_{V_1} = f_P \cup f_T \cup f_I \\
f'_P &: N \to N \quad \text{by } f'_P = id_N \\
f'_T &: E_{IG} \to E_{I_2} \quad \text{with } E_{IG} = E_{I_{10p}} \cup E_{I_{p2t}} \cup E_{I_{2p}} \quad \text{by} \\
& f'_T = (f'_T(p), f'_T(p)) \quad \text{for } (p, t) \in E_{I_{10p}} \\
& f'_T = (f'_T(p), f'_T(t)) \quad \text{for } (p, t) \in E_{I_{p2t}} \\
& f'_T = (f'_T(t), f'_T(p)) \quad \text{for } (t, p) \in E_{I_{2p}} \\
f'_E &: E_{NA} \to E_{\text{INA}} \quad \text{with } E_{INA} = E_{in} \cup E_{out} \quad \text{by} \\
& f'_E = (f_{E_{I_1}}(t, n, i), f_{E_{I_2}}(t, n, i)) \quad \text{for } (t, n, i) \in E_{IN} \cup E_{out} \land i \in \{in, out\} \\
f'_E &: E_{IAE} \to E_{\text{EA}} \quad \text{with } E_{\text{IAE}} = E_{\text{in}} \cup E_{\text{out}} \quad \text{by} \\
& f'_E = (f_{E_{I_1}}(t, p, n), f_{E_{I_2}}(t, p, n)) \quad \text{for } (t, p, n) \in E_{\text{IAE}} \\
& f'_E = (f_{E_{I_1}}(t, p, n), f_{E_{I_2}}(t, p, n)) \quad \text{for } (t, p, n) \in E_{\text{IAE}} \\
& f'_E = (f_{E_{I_1}}(t, p, n), f_{E_{I_2}}(t, p, n)) \quad \text{for } (t, p, n) \in E_{\text{IAE}} \\
& f'_E = (f_{E_{I_1}}(t, p, n), f_{E_{I_2}}(t, p, n)) \quad \text{for } (t, p, n) \in E_{\text{IAE}}.
\end{align*}
\]
Lemma 1 (Well-Definedness of Morphism Translation)

For each \( f : NI_1 \to NI_2 \) in PTINet with \( f \in \mathcal{M}_1 \) is \( \mathcal{F} (f) : \mathcal{F} (NI_1) \to \mathcal{F} (NI_2) \) in AGraphs\textsubscript{PNTG} well-defined with \( \mathcal{F} (f) \in \mathcal{M}_2 \). Moreover \( \mathcal{F} \) preserves inclusions.

Proof.

A detailed proof is given in [MEE11] showing the following steps:

1. \( f'_V, f'_T, f'_E, f'_E \text{Ext}, f'_E \text{Out} \) are well-defined w.r.t. codomain.
2. The components of \( \mathcal{F}(f) \) are compatible with sources and targets.
3. The components of \( \mathcal{F}(f) \) are compatible with typing morphisms.
4. \( f \in \mathcal{M}_1 \) (inclusion) implies \( \mathcal{F}(f) \in \mathcal{M}_2 \) (inclusion).

\( \square \)

5.5 Translation of Pushouts

We have to show, that if (1) is a PO in PTINet with \( f_i \in \mathcal{M}_1 \), then we have that (2) is a PO in AGraphs\textsubscript{PNTG} with \( \mathcal{F}(f_i) \in \mathcal{M}_2 \).

\[ \begin{array}{ccc}
NI_0 \xrightarrow{f_i} & NI_1 \\
\text{f}_2 \downarrow & \downarrow \text{f}_4 \\
NI_2 \xrightarrow{f_i} & NI_3
\end{array} \]

Since POs in PTINet are constructed componentwise, we know that the \( P \)-, \( T \)- and \( I \)-components of (1) are POs in Sets. Since also POs in AGraphs\textsubscript{ATG} and AGraphs\textsubscript{PNTG} are constructed componentwise we have to show that the \( V_G \)-, \( V_D \)-, \( E_G \)-, \( E_NA \)- and \( E_{EA} \)-components of (2) are POs in Sets. This is clear for the \( V_G \)-components \( f_{V_G} = f_{IP} \circ f_{IT} \) because POs are compatible with coproducts and for \( f_{ID} \), because all components are identities. For the \( E_G \)-component we have to show that (3) is PO in Sets, which follows if (4) and similar (5) resp. (6) with “to2p” and “\( f_{IT} \times f_{IP} \)” replaced by “\( 2pIT \)” and “\( f_{IP} \times f_{IT} \)” resp. “\( 2pP \)” and “\( f_{IT} \times f_{IP} \)” are POs.

\[ \begin{array}{ccc}
E_{0G} \xrightarrow{\mathcal{F}(f_i)_G} & E_{1G} \\
\mathcal{F}(f_2)_G \downarrow & \downarrow \mathcal{F}(f_4)_G \\
E_{2G} \xrightarrow{\mathcal{F}(f_i)_G} & E_{3G}
\end{array} \]

\[ \begin{array}{ccc}
E_{002p} \xrightarrow{f_{IT} \times f_{IP}} & E_{102p} \\
f_{2T} \circ f_{SP} \downarrow & \downarrow f_{4T} \circ f_{SP} \\
E_{202p} \xrightarrow{f_{IT} \times f_{SP}} & E_{302p}
\end{array} \]

For the \( E_{NA} \)- and \( E_{EA} \)-components, it is sufficient to show POs (5) and (6) and similar (5a) with “in” replaced by “out” and (6a) with “pre” replaced by “post”.

\[ \begin{array}{ccc}
E_{0i} \xrightarrow{f_{IT} \times id_{i}} & E_{1i} \\
f_{2T} \times id_{i} \downarrow & \downarrow f_{2T} \times id_{i} \\
E_{2i} \xrightarrow{f_{IT} \times id_{i}} & E_{3i}
\end{array} \]

\[ \begin{array}{ccc}
E_{0w} \xrightarrow{f_{IT} \times f_{SP}} & E_{1w} \\
f_{2T} \times f_{ST} \downarrow & \downarrow f_{2T} \times f_{ST} \\
E_{2w} \xrightarrow{f_{IT} \times f_{SP}} & E_{3w}
\end{array} \]
All these diagrams commute, because each product component commutes by assumption. But it is more difficult to show explicitly, that they are POs (see for example Lemma 2 below), because products of POs are in general not POs. An example is the translation of the PO in \textbf{PTINet} shown in Figure 2 to the PO in \textbf{AGraphs}_{\text{NTG}} shown in Figure 4.

**Lemma 2** (Translation of Pushouts)

*Diagrams (4) and (4a) are pushouts.*

*Proof.* See [MEE11].

### 5.6 Creation of Injective Morphisms

Given \(\mathcal{F}(NI_1), \mathcal{F}(NI_2)\) and \(f' : \mathcal{F}(NI_1) \to \mathcal{F}(NI_2) \in \mathcal{M}_2\) with type compatible morphisms

- \(f'_{G_v} : V_{1G} \to V_{2G}\) with \(V_{1G} = P_1 \square T_i \square I_i\) for \(i = 1, 2\)
- \(f'_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}\) with \(f'_{\mathbb{N}} = id_{\mathbb{N}}\)
- \(f'_{E_G} : E_{1G} \to E_{2G}\) with \(E_{1G} = E_{in2p} \square E_{ip2r} \square E_{ir2p}\)
- \(f'_{E_{NA}} : E_{1NA} \to E_{2NA}\) with \(E_{1NA} = E_{in} \square E_{out}\)
- \(f'_{E_{EA}} : E_{1EA} \to E_{2EA}\) with \(E_{1EA} = E_{ipw_{pre}} \square E_{iw_{post}}\)

Define \(f : NI_1 \to NI_2\) with \(NI_j = (P_j, T_j, \text{pre}_j, \text{post}_j, I_j, m_j)\) for \(j = 1, 2\) by \(f = (f_P : P_1 \to P_2, f_T : T_1 \to T_2, f_I : I_1 \to I_2)\) with

- \(f_T(t) = f'_{V_G}(t)\) for \(t \in T_1 \subseteq V_{1G}\)
- \(f_P(p) = f'_{V_G}(p)\) for \(p \in P_1 \subseteq V_{1G}\)
- \(f_I(x) = f'_{V_G}(x)\) for \(x \in I_1 \subseteq V_{1G}\)

Well-definedness of \(f : NI_1 \to NI_2 \in \mathcal{M}_1\) follows from Lemma 3 below, where the proof of part 2 is based on Lemma 4. The proofs of both Lemma are given in [MEE11].

**Lemma 3** (Well-Definedness of Creation of Injective Morphisms)

*Given the construction above for \(f : NI_1 \to NI_2\). The following holds:*

1. \(f'_{V_G}(t) \in T_2, f'_{V_G}(p) \in P_2, f'_{V_G}(x) \in I_2,\) and
2. squares (1), (2) to the right commute with injective \(f_P, f_T, f_I\).

**Lemma 4** (PTI-Morphism-Lemma)

\(f : NI_1 \to NI_2\) is an injective \textbf{PTINet}-morphism \(\iff f = (f_P, f_T, f_I)\) is injective with 1–4, where

1. \(\forall t \in T_1, p \in \bullet t \iff f_P(p) \in \bullet f_T(t)\) and \(\forall t \in T_1, p \in \bullet t \iff f_P(p) \in f_T(t)\)
2. \(\forall (p, t) \in P_1 \square T_1 = E_{1p2r}, \ (t, p, n) \in E_{1w_{pre}} \iff (f_P(p), f_T(t), n) \in E_{2w_{pre}}\) and \(\forall (t, p) \in T_1 \square P_1 = E_{1p2r}, \ (t, p, n) \in E_{1w_{post}} \iff (f_T(t), f_P(p), n) \in E_{2w_{post}}\)
Translation and Creation of Transformations

3. \( \forall t \in T_1, \quad \text{card} (\bullet t) = n \iff \text{card} (\bullet f_T (t)) = n \quad \text{and} \quad \text{card} (\bullet \circ t) = n \iff \text{card} (f_T (t) \circ \bullet) = n \)
   \[ \bullet t = \{ p \in P_1 \mid \text{pre}_1 (t) (p) > 0 \} \quad \text{and} \quad \circ t = \{ p \in P_1 \mid \text{post}_1 (t) (p) > 0 \} \]

4. \( \forall x \in I_1, (x, p) \in E_{1 \to 2 p} \iff (f_I (x), f_P (p)) \in E_{2 \to 2 p} \)

5.7 Preservation of Initial Pushouts

The proof of this property is based on the initial PO constructions for PTINet in [MGE+09] and for AGraphsATG in [EEPT06]. Details of the proof are given in [MEE11]. An example is given in Figure 5, where (1) is an initial PO over \( f \) in PTINet, (2) the induced PO over \( \mathcal{F} (f) \), and the initial PO over \( \mathcal{F} (f) \) in AGraphsPTG is given by the outer diagram with corners \( B', C', \mathcal{F} (L), \mathcal{F} (G) \). Since \( i' \) and \( j' \) are isomorphisms, diagram (2) is already initial PO over \( \mathcal{F} (f) \).

![Figure 5: Preservation of Initial Pushouts](image)

6 Conclusion and Future Work

As pointed out already in Section 1 we want to develop a general framework to establish a formal relationship between different \( \mathcal{M} \)-adhesive transformation systems based on \( \mathcal{M} \)-adhesive categories. The main idea is to construct a suitable \( \mathcal{M} \)-functor between the corresponding \( \mathcal{M} \)-adhesive categories, which translates pushouts, creates morphisms and preserves initial pushouts.
This allows by Theorem 1 and Theorem 2 the translation and creation of transformations between the corresponding $\mathcal{M}$-adhesive transformation systems, including parallel and sequential independence of transformations. Moreover, we have discussed the restriction to injective matches via $\mathcal{M}_1$-morphisms, which requires only a functor for $\mathcal{M}_1$-morphisms.

In Section 5 we have discussed a corresponding functor from Petri nets with individual tokens to typed attributed graphs. We have verified that this functor translates pushouts of $\mathcal{M}_1$-morphisms, creates $\mathcal{M}_1$-morphisms and preserves initial pushouts over $\mathcal{M}_1$-morphisms, which allows the application of Theorem 1 and Theorem 2 in connection with Remark 4.

In future work, we will provide sufficient conditions in order to ensure that the $\mathcal{M}$—functor preserves initial pushouts. In the long run, this should allow the analysis of interesting properties of Petri net transformation systems, like termination and local confluence in addition to parallel and sequential independence, 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, e.g. high-level Petri nets and typed attributed graphs as well as triple graphs and flattening of triple graphs.

Acknowledgements: Some of the authors are partly supported by the German Research Council project Behaviour Simulation and Equivalence of Systems Modelled by Graph Transformation (Behaviour-GT).

References


2 First steps in this direction are Lemma 5 and Lemma 6 in [MEE11].
Translation and Creation of Transformations


