View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Electronic Communications of the EASST (European Association of Software Science and...

Electronic Communications of the EASST
Volume 58 (2013)

Proceedings of the 12th International Workshop on Graph Transformation and Visual Modeling Techniques (GTVMT 2013)

Analysis of Hypergraph Transformation Systems in AGG based on $\mathscr{M}$-Functors

Maria Maximova, Hartmut Ehrig and Claudia Ermel

13 pages

Guest Editors: Matthias Tichy, Leila Ribeiro
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Analysis of Hypergraph Transformation Systems in AGG based on $\mathscr{M}$-Functors

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: Hypergraph transformation systems are examples of $\mathscr{M}$-adhesive transformation systems based on $\mathscr{M}$-adhesive categories. For typed attributed graph transformation systems, the tool environment AGG allows the modelling, the simulation and the analysis of graph transformations. A corresponding tool for analysis of hypergraph transformation systems does not exist up to now. The purpose of this paper is to establish a formal relationship between the corresponding $\mathscr{M}$-adhesive transformation systems, which allows us the translation of hypergraph transformations into typed attributed graph transformations with equivalent behavior, and, vice versa, the creation of hypergraph transformations from typed attributed graph transformations. This formal relationship is based on the general theory of $\mathscr{M}$-functors between different $\mathscr{M}$-adhesive transformation systems. We construct a functor between the $\mathscr{M}$-adhesive categories of hypergraphs and of typed attributed graphs, and show that our construction yields an $\mathscr{M}$-functor with suitable properties. We then use existing results for $\mathscr{M}$-functors to show that analysis results for hypergraph transformation systems can be obtained using AGG by analysis of the translated typed attributed graph transformation system. This is shown in general and for a concrete example.

Keywords: $\mathscr{M}$-adhesive transformation system, graph transformation, hypergraph transformation, $\mathscr{M}$-adhesive category, $\mathscr{M}$-functor, critical pair analysis, AGG

1 Introduction

In the theory of graph transformation, various related approaches exist. *Hypergraphs* have shown to be appropriate e.g. for evaluation of functional expressions since they allow a function with n arguments to be modelled by an hyperedge with one source node and n target nodes [Plu93]. Other applications are distributed systems [BCK02] and diagram representation [Min00].

Hypergraph transformation is related to algebraic graph transformation [EEPT06], where structural changes are modelled in the double-pushout (DPO) approach for the category of (typed, attributed) graphs, which has been generalised to $\mathscr{M}$-adhesive categories, relying on a class $\mathscr{M}$ of monomorphisms. The DPO approach is a suitable description of transformations leading to results like the Local Church-Rosser, Parallelism, Concurrency, and Local Confluence Theorems [EEPT06]. The well-established tool AGG [AGG12] supports modelling and analysis of (typed, attributed) graph transformation systems. However, up to now there exists no tool support for directly analysing confluence of hypergraph transformation systems.

In our previous paper [MEE11], we have proposed formal criteria ensuring a semantical correspondence of reconfigurable Petri nets and their corresponding representations as graph transformation systems. The aim of our previous work was to establish a formal basis allowing us to translate Petri net transformations into graph transformations and, vice versa, to create Petri net transformations from graph transformations such that the behavior of Petri net transformations can be simulated by simulating their translation using the graph transformation tool AGG.

In [MEE11], we established the new framework of $\mathscr{M}$-functors $\mathscr{F} : (\mathbf{C_1}, \mathscr{M}_1) \to (\mathbf{C_2}, \mathscr{M}_2)$ between $\mathscr{M}$-adhesive categories. This framework allows to translate transformations in $(\mathbf{C_1}, \mathscr{M}_1)$ into corresponding transformations in $(\mathbf{C_2}, \mathscr{M}_2)$ and, vice versa, to create transformations in $(\mathbf{C_1}, \mathscr{M}_1)$ from those in $(\mathbf{C_2}, \mathscr{M}_2)$.

Building on this previous work, we have extended this framework in [MEE12] to allow the analysis of interesting properties like termination, local confluence and functional behavior, in addition to parallel and sequential independence, using the corresponding results and analysis tools like AGG for graph transformation systems.

Hence, in this paper we define an $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}} : (\mathbf{C_1}, \mathscr{M}_1) \to (\mathbf{C_2}, \mathscr{M}_2)$, where $(\mathbf{C_1}, \mathscr{M}_1)$ are hypergraphs and $(\mathbf{C_2}, \mathscr{M}_2)$ typed attributed graphs. In our main results, we show that the functor $\mathscr{F}_{\mathrm{HG}}$ satisfies all the properties required by the general framework to guarantee the transfer, i.e., translation and creation of transformations and local confluence. This allows us in particular to apply the well-known critical pair analysis for typed attributed graph transformations supported by the AGG-tool [AGG12] to analyse these properties for hypergraph transformations. In contrast to previous instantiations of $\mathscr{M}$-functors in [MEE11], we do not have to restrict our functor to injective morphisms.

The paper is structured as follows: Section 2 introduces the basic notions of $\mathscr{M}$-adhesive transformation systems and $\mathscr{M}$-functors to define a formal relationship between two different $\mathscr{M}$-adhesive categories. In Section 3, we construct the functor $\mathscr{F}_{\mathrm{HG}}$ between hypergraph and typed attributed graph transformation systems, and show that this functor satisfies the properties of $\mathscr{M}$-functors and some additional properties that are required in the general theory. In Section 4, we study the $\mathscr{F}$-transfer of local confluence by analysing $\mathscr{F}$-reachable critical pairs and show that the $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}}$ from Section 3 satisfies the required properties. The result is used in Section 5 to analyse a hypergraph transformation system using AGG on the functorial translation of the system. In Section 6, we compare our approach to related work, conclude the paper and give an outlook to future research directions. Detailed proofs are given in [MEE13].

2 $\mathscr{M}$-Adhesive Categories, Transformation Systems, $\mathscr{M}$-Functors

In this section we concentrate on some basic concepts and results that are important for our approach and which we review from our previous paper [MEE11]. Our considerations are based on the framework of $\mathscr{M}$-adhesive categories. An $\mathscr{M}$-adhesive category [EGH10], consists of a category $\mathbf{C}$ together with a class $\mathscr{M}$ of monomorphisms such that the following properties hold: $\mathbf{C}$ has pushouts (POs) and pullbacks (PBs) along $\mathscr{M}$-morphisms, $\mathscr{M}$ is closed under isomorphisms, composition, decomposition, POs and PBs, and POs along $\mathscr{M}$-morphisms are $\mathscr{M}$-VK-squares (see Figure 1), i.e., the VK-property holds for all commutative cubes, where the given PO with $m \in \mathscr{M}$ is in the bottom, the back faces are PBs and all vertical morphisms a, b, c and d are in

$\mathscr{M}$. The VK-property means that the top face is a PO iff the front faces are PBs.

The concept of $\mathscr{M}$-adhesive categories generalises that of adhesive [LS04], adhesive HLR, and weak adhesive HLR categories [EEPT06]. The categories of typed attributed graphs, hypergraphs and several categories of Petri nets are weak adhesive HLR (see [EEPT06]) and hence also $\mathscr{M}$-adhesive. A set of transformation rules in an $\mathscr{M}$-adhesive category constitutes an $\mathscr{M}$-adhesive transformation system [EGH10].

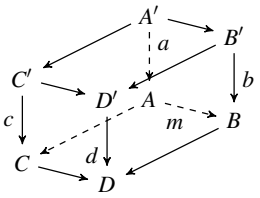


Figure 1: $\mathscr{M}$-VK-square

Definition 1 ($\mathscr{M}$-Adhesive Transformation System)
Given an $\mathscr{M}$-adhesive category $(\mathbf{C},\mathscr{M})$, an $\mathscr{M}$-adhesive transformation system $AS=(\mathbf{C},\mathscr{M},P)$ has a set P of productions of the form $\rho=(L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ with $l,r\in\mathscr{M}$. A direct transformation $G \overset{\rho,m}{\Longrightarrow} H$ via ρ and match m consists of two pushouts according to the DPO approach [EEPT06].

We use the notion of an $\mathscr{M}$-functor [MEE11] to define a formal relationship between two different $\mathscr{M}$-adhesive transformation systems.

Definition 2 ($\mathscr{M}$-Functor)
A functor $\mathscr{F}:(\mathbf{C_1},\mathscr{M}_1)\to(\mathbf{C_2},\mathscr{M}_2)$ between $\mathscr{M}$-adhesive categories is called *$\mathscr{M}$-functor* if $\mathscr{F}(\mathscr{M}_1)\subseteq\mathscr{M}_2$ and $\mathscr{F}$ preserves pushouts along $\mathscr{M}$-morphisms.

Given an $\mathscr{M}$-adhesive transformation system $AS_1=(\mathbf{C_1},\mathscr{M}_1,P_1)$, we want to translate transformations from AS_1 to $AS_2=(\mathbf{C_2},\mathscr{M}_2,P_2)$ with translated productions $P_2=\mathscr{F}(P_1)$ and, vice versa, we want to create transformations in AS_1 from the corresponding transformations in AS_2. This can be handled by Theorem 1 below, shown in [MEE11].

By definition, each $\mathscr{M}$-functor $\mathscr{F}:(\mathbf{C_1},\mathscr{M}_1)\to(\mathbf{C_2},\mathscr{M}_2)$ translates each production $\rho=(L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ in P_1 with $l,r\in\mathscr{M}_1$ into $\mathscr{F}(\rho)=(\mathscr{F}(L) \overset{\mathscr{F}(l)}{\leftarrow} \mathscr{F}(K) \overset{\mathscr{F}(r)}{\rightarrow} \mathscr{F}(R))$ in $P_2=\mathscr{F}(P_1)$ with $\mathscr{F}(l),\mathscr{F}(r)\in\mathscr{M}_2$ and each direct transformation $G \overset{\rho,m}{\Longrightarrow} H$ in AS_1 given by DPO $(1)+(2)$ into a direct transformation $\mathscr{F}(G) \overset{\mathscr{F}(\rho),\mathscr{F}(m)}{\Longrightarrow} \mathscr{F}(H)$ in AS_2 given by DPO $(3)+(4)$.

$$\begin{array}{ccccc} L & \overset{l}{\leftarrow} & K & \overset{r}{\rightarrow} & R \\ {\scriptstyle m}\downarrow & (1) & \downarrow & (2) & \downarrow \\ G & \leftarrow & D & \rightarrow & H \end{array} \quad\Rightarrow\quad \begin{array}{ccccc} \mathscr{F}(L) & \overset{\mathscr{F}(l)}{\leftarrow} & \mathscr{F}(K) & \overset{\mathscr{F}(r)}{\rightarrow} & \mathscr{F}(R) \\ {\scriptstyle \mathscr{F}(m)}\downarrow & (3) & \downarrow & (4) & \downarrow \\ \mathscr{F}(G) & \leftarrow & \mathscr{F}(D) & \rightarrow & \mathscr{F}(H) \end{array}$$

Vice versa, we say $\mathscr{F}$ creates direct transformations, if for each direct transformation $\mathscr{F}(G) \overset{\mathscr{F}(\rho),m'}{\Longrightarrow} H'$ in AS_2 there is a direct transformation $G \overset{\rho,m}{\Longrightarrow} H$ in AS_1 with $\mathscr{F}(m)=m'$ and $\mathscr{F}(H)\cong H'$ leading to $\mathscr{F}(G) \overset{\mathscr{F}(\rho),\mathscr{F}(m)}{\Longrightarrow} \mathscr{F}(H)$ in AS_2. In the following, we provide two conditions in order to show creation of direct transformations and transformations, i.e., sequences of direct transformations written $G \overset{*}{\Rightarrow} H$.

Theorem 1 (Translation and Creation of Transformations)
Each $\mathscr{M}$-functor $\mathscr{F}:(\mathbf{C_1},\mathscr{M}_1)\to(\mathbf{C_2},\mathscr{M}_2)$ translates (direct) transformations.
Vice versa, $\mathscr{F}$ creates (direct) transformations if we have the following two conditions:

- *($\mathscr{F}$ **creates morphisms**): For all $m' : \mathscr{F}(L) \to \mathscr{F}(G)$ in $(\mathbf{C_2}, \mathscr{M}_2)$, there is exactly one morphism $m : L \to G$ with $\mathscr{F}(m) = m'$.*
- *($\mathscr{F}$ **preserves initial pushouts**): $(\mathbf{C_1}, \mathscr{M}_1)$ has initial pushouts and for each initial pushout (1) over $m : L \to G$, also (2) is initial pushout over $\mathscr{F}(m)$ in $(\mathbf{C_2}, \mathscr{M}_2)$.*

$$\begin{array}{ccc} B & \xrightarrow{b} & L \\ \downarrow & (1) & \downarrow m \\ C & \longrightarrow & G \end{array} \quad \Rightarrow \quad \begin{array}{ccc} \mathscr{F}(B) & \xrightarrow{\mathscr{F}(b)} & \mathscr{F}(L) \\ \downarrow & (2) & \downarrow \mathscr{F}(m) \\ \mathscr{F}(C) & \longrightarrow & \mathscr{F}(G) \end{array}$$

The proof for Theorem 1 is given in [MEE11]. Moreover, it is shown under the same assumptions that $\mathscr{F}$ translates and creates parallel and sequential independence of transformations. Concerning the definition and the role of initial pushouts for the applicability of productions we refer to [EEPT06, MEE11].

3 $\mathscr{M}$-Functor from Hypergraphs to Typed Attributed Graphs

Our aim is to construct a functor from hypergraphs to typed attributed graphs to be able to analyse hypergraphs by anlysing typed attributed graphs according to the general theory from [MEE11] and [MEE12]. For this purpose, we review on the one hand the category $(\mathbf{HyperGraphs}, \mathscr{M}_1)$ of hypergraphs with the class $\mathscr{M}_1$ of all injective hypergraph morphisms, which is shown to be $\mathscr{M}$-adhesive in [EEPT06]. On the other hand, we review the category of typed attributed graphs $(\mathbf{AGraphs}_{\mathbf{ATG}}, \mathscr{M}_2)$ with the class $\mathscr{M}_2$ of all injective typed attributed graph morphisms, which is also shown to be $\mathscr{M}$-adhesive in [EEPT06], and we define a suitable attributed hypergraph type graph $ATG = HGTG$. Moreover, we construct a functor $\mathscr{F}_{\mathrm{HG}}$ between both categories and show that the general result from Theorem 1 is applicable to this functor.

Definition 3 (Category **HyperGraphs** [EEPT06])
A hypergraph G is defined as $G = (V_G, E_G, s_G, t_G)$, where V_G is a set of hypergraph nodes, E_G is a set of hyperedges and $s_G, t_G : E_G \to V_G^*$ are functions assigning the string $s_G(e)$ of source nodes resp. $t_G(e)$ of target nodes to each hyperedge e.
Consider two hypergraphs $G_1 = (V_{G_1}, E_{G_1}, s_{G_1}, t_{G_1})$ and $G_2 = (V_{G_2}, E_{G_2}, s_{G_2}, t_{G_2})$. A hypergraph morphism $f : G_1 \to G_2$ is given by a tuple of functions $f = (f_V : V_{G_1} \to V_{G_2}, f_E : E_{G_1} \to E_{G_2})$ such that the diagram to the right commutes with source and target functions, i.e., $s_{G_2} \circ f_E = f_V^* \circ s_{G_1}$ and $t_{G_2} \circ f_E = f_V^* \circ t_{G_1}$, where $f_V^* : V_{G_1}^* \to V_{G_2}^*$ with $\lambda \mapsto \lambda$ and $x_1 \ldots x_n \mapsto f_V(x_1) \ldots f_V(x_n)$.

$$\begin{array}{ccc} E_{G_1} & \overset{s_{G_1}}{\underset{t_{G_1}}{\rightrightarrows}} & V_{G_1}^* \\ {\scriptstyle f_E}\downarrow & = & \downarrow{\scriptstyle f_V^*} \\ E_{G_2} & \overset{s_{G_2}}{\underset{t_{G_2}}{\rightrightarrows}} & V_{G_2}^* \end{array}$$

According to [EEPT06], the category $(\mathbf{HyperGraphs}, \mathscr{M}_1)$ of hypergraphs with a class $\mathscr{M}_1$ of all injective morphisms is $\mathscr{M}$-adhesive, where pushouts are constructed componentwise.
Attributed graphs and morphisms between them form the category **AGraphs**, where each object is a pair (G, D) of an E-graph G with signature E (shown to the right) and Σ-*nat* algebra D, where in the following we only use $D = T_{\Sigma\text{-}nat} \cong NAT$ (with the term algebra $T_{\Sigma\text{-}nat}$ and the ordinary natural numbers algebra NAT). This means, G is given by $G = (V_G^G, V_D^G, E_G^G, E_{NA}^G, E_{EA}^G, (s_j^G, t_j^G)_{j \in \{G,NA,EA\}})$, where V_G^G resp. V_D^G are the graph resp. data

$$\begin{array}{ccccc} & & E_G & \overset{s_G}{\underset{t_G}{\rightrightarrows}} & V_G \\ & {\scriptstyle s_{EA}}\nearrow & & & \nwarrow{\scriptstyle s_{NA}} \\ E_{EA} & \xrightarrow[t_{EA}]{} & V_D & \xleftarrow[t_{NA}]{} & E_{NA} \end{array}$$

nodes of G, E_G^G, E_{NA}^G resp. E_{EA}^G are the graph edges resp. node attribute and edge attribute edges of G and s_j^G, t_j^G are corresponding source and target functions for the edges.

The notion of attributed graphs combined with the typing concept leads to the well-known category of typed attributed graphs $\mathbf{AGraphs_{ATG}}$, where attributed graphs are typed over an attributed type graph ATG [EEPT06]. Here, we consider a specific type graph $HGTG$ to express hypergraphs as typed attributed graphs, which is shown in Figure 2. [1]

The meaning of every depicted element of $HGTG$ is as follows: Nodes of type *Node* and *Edge* represent hypergraph nodes and hyperedges. Edges of types *n2e*, *e2n* represent hyperedge tentacles and are attributed by a number *num* which contains the position of a node in the source (resp. target) string of the considered hyperedge. Nodes of type *Edge* have two attributes *in* and *out* giving the number of nodes in the pre- and postdomain of a hyperedge (to ensure the preservation of an *Edge* node's environment using typed attributed graph morphisms). All node and edge attributes are typed over natural numbers.

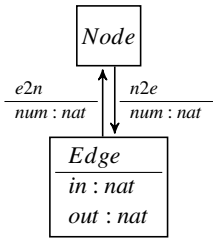


Figure 2: Attributed type graph $HGTG$

We consider the category $\mathbf{AGraphs_{HGTG}}$ with fixed data type NAT and identical algebra homomorphism, which implies that the V_D-component of morphisms is the identity.

According to [EEPT06], the category $(\mathbf{AGraphs_{ATG}}, \mathscr{M})$ is $\mathscr{M}$-adhesive for each type graph ATG, where $\mathscr{M}$-morphisms are injective with isomorphic data type part. Hence also the special case of $(\mathbf{AGraphs_{ATG}}, \mathscr{M})$ with $ATG = HGTG$ is $\mathscr{M}$-adhesive. The subcategory $(\mathbf{AGraphs_{HGTG}}, \mathscr{M}_2)$ with identical algebra homomorphism as considered above is also $\mathscr{M}$-adhesive for the subclass $\mathscr{M} = \mathscr{M}_2$ of all injective typed attributed graph morphisms.

We are using the $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathscr{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathscr{M}_2)$ defined below for the translation of **HyperGraphs** objects and morphisms into the corresponding $\mathbf{AGraphs_{HGTG}}$ objects and morphisms.

Definition 4 ($\mathscr{M}$-Functor $\mathscr{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathscr{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathscr{M}_2)$)
Consider a hypergraph $G = (V_G, E_G, s_G, t_G)$. We define the object $\mathscr{F}_{\mathrm{HG}}(G) = ((G', NAT), type)$[2] in $\mathbf{AGraphs_{HGTG}}$ with morphism $type : (G', NAT) \to (HGTG, D_{fin})$ and E-graph $G' = (V_G^{G'}, V_D^{G'} = \mathbb{N}, E_G^{G'}, E_{NA}^{G'}, E_{EA}^{G'}, (s_j^{G'}, t_j^{G'})_{j \in \{G,NA,EA\}})$ as follows:

$$
\begin{aligned}
V_G^{G'} &= V_G \uplus E_G \text{ (graph nodes)} \\
E_G^{G'} &= E_{n2e}^{G'} \uplus E_{e2n}^{G'} \text{ (graph edges) with} \\
&\quad E_{n2e}^{G'} = \{(v,e,n) \in V_G \times E_G \times \mathbb{N} \mid s_G^n(e) = v\}, \\
&\quad E_{e2n}^{G'} = \{(e,v,n) \in E_G \times V_G \times \mathbb{N} \mid t_G^n(e) = v\}, \\
&\quad \text{where } s_G^n(e) \text{ is the } n\text{-th node in the string } s_G(e) \text{ and similar for } t_G^n(e), \\
E_{NA}^{G'} &= E_{in}^{G'} \uplus E_{out}^{G'} \text{ (node attribute edges) with} \\
&\quad E_{in}^{G'} = \{(e,n,in) \mid (e,n) \in E_G \times \mathbb{N} \wedge |s_G(e)| = n\},
\end{aligned}
$$

[1] Node and edge attributes are depicted in compact notation as node/edge inscriptions together with their data type.
[2] In the following, we also use the short notation $\mathscr{F}_{\mathrm{HG}}(G) = G'$.

$$E^{G'}_{out} = \{(e,n,out) \mid (e,n) \in E_G \times \mathbb{N} \wedge |t_G(e)| = n\}, \text{ where } |w| \text{ is the length of } w,$$
$$E^{G'}_{EA} = E^{G'}_s \uplus E^{G'}_t \text{ (edge attribute edges) with}$$
$$E^{G'}_s = \{(n,v,e) \in \mathbb{N} \times V_G \times E_G \mid s^n_G(e) = v\}^3,$$
$$E^{G'}_t = \{(n,e,v) \in \mathbb{N} \times E_G \times V_G \mid t^n_G(e) = v\}^3,$$
(and the corresponding source and target functions:)
$$s^{G'}_G, t^{G'}_G : E^{G'}_G \to V^{G'}_G \text{ defined by } s^{G'}_G(x,y,n) = x,\ t^{G'}_G(x,y,n) = y,$$
$$s^{G'}_{NA} : E^{G'}_{NA} \to V^{G'}_G,\ t^{G'}_{NA} : E^{G'}_{NA} \to \mathbb{N} \text{ defined by } s^{G'}_{NA}(e,n,x) = e,\ t^{G'}_{NA}(e,n,x) = n,$$
$$s^{G'}_{EA} : E^{G'}_{EA} \to E^{G'}_G,\ t^{G'}_{EA} : E^{G'}_{EA} \to \mathbb{N} \text{ defined by } s^{G'}_{EA}(n,x,y) = (x,y,n),\ t^{G'}_{EA}(n,x,y) = n.$$

The $\mathbf{AGraphs_{HGTG}}$-morphism $type : (G', NAT) \to (HGTG, D_{fin})$ is given by the final morphism of data types from NAT to the final algebra D_{fin} and $type^{G'} : G' \to HGTG$ is given by E-graph morphism $type^{G'} = (type_{V_G}, type_{V_D}, type_{E_G}, type_{E_{NA}}, type_{E_{EA}})$, where each component is mapped to the obvious type in the type graph $HGTG$, e.g., $type_{V_G} : V^{G'}_G \to V^{HGTG}_G$ with $x \mapsto Node$ (if $x \in V_G$), $x \mapsto Edge$ (if $x \in E_G$).

For each hypergraph morphism $f : G_1 \to G_2$ with $f = (f_V : V_{G_1} \to V_{G_2}, f_E : E_{G_1} \to E_{G_2})$, we define $\mathscr{F}_{\mathrm{HG}}(f) : \mathscr{F}_{\mathrm{HG}}(G_1) \to \mathscr{F}_{\mathrm{HG}}(G_2)$, where in short notation $\mathscr{F}_{\mathrm{HG}}(G_i) = (V^{G_i}_G, \mathbb{N}, E^{G_i}_G, E^{G_i}_{NA}, E^{G_i}_{EA}, (s^{G_i}_j, t^{G_i}_j)_{j \in \{G,NA,EA\}})$ with $i \in \{1,2\}$ by $\mathscr{F}_{\mathrm{HG}}(f) = f' = (f'_{V_G}, f'_{V_D} = id_{\mathbb{N}}, f'_{E_G}, f'_{E_{NA}}, f'_{E_{EA}})$ with

$$f'_{V_G} : V^{G_1}_G \to V^{G_2}_G \text{ with } V^{G_i}_G = V_{G_i} \uplus E_{G_i} \text{ for } i \in \{1,2\} \text{ by } f'_{V_G} = f_V \uplus f_E$$
$$f'_{E_G} : E^{G_1}_G \to E^{G_2}_G \text{ with } E^{G_i}_G = E^{G_i}_{n2e} \uplus E^{G_i}_{e2n} \text{ for } i \in \{1,2\} \text{ by}$$
$$f'_{E_G}(v,e,n) = (f_V(v), f_E(e), n) \text{ for } (v,e,n) \in E^{G_1}_{n2e}$$
$$f'_{E_G}(e,v,n) = (f_E(e), f_V(v), n) \text{ for } (e,v,n) \in E^{G_1}_{e2n}$$
$$f'_{E_{NA}} : E^{G_1}_{NA} \to E^{G_2}_{NA} \text{ with } E^{G_i}_{NA} = E^{G_i}_{in} \uplus E^{G_i}_{out} \text{ for } i \in \{1,2\} \text{ by}$$
$$f'_{E_{NA}}(e,n,x) = (f_E(e), n, x) \text{ for } (e,n,x) \in E^{G_1}_{in} \uplus E^{G_1}_{out} \wedge x \in \{in, out\}$$
$$f'_{E_{EA}} : E^{G_1}_{EA} \to E^{G_2}_{EA} \text{ with } E^{G_i}_{EA} = E^{G_i}_s \uplus E^{G_i}_t \text{ for } i \in \{1,2\} \text{ similar}^3 \text{ to } f'_{E_G} : E^{G_1}_G \to E^{G_2}_G$$

An example for using the functor $\mathscr{F}_{\mathrm{HG}}$ on objects and morphisms is shown in Figure 3, where the typed attributed graphs on the right together with the morphism between them are the translation of the corresponding hypergraphs and the morphism on the left. As usual in the hypergraph notation, only the target nodes of a hyperedge are marked by arrows.

Note that $\mathscr{F}_{\mathrm{HG}}$ defined above is a well-defined $\mathscr{M}$-functor in the sense of Definition 4. This includes that the components of $\mathscr{F}_{\mathrm{HG}}(f)$ are well-defined w.r.t. their codomain and that they are compatible with source and target functions as well as typing morphisms. $\mathscr{F}_{\mathrm{HG}}$ is a functor, because $\mathscr{F}_{\mathrm{HG}}$ preserves identities and composition. Moreover, $\mathscr{F}_{\mathrm{HG}}$ is an $\mathscr{M}$-functor, because we have $\mathscr{F}_{\mathrm{HG}}(\mathscr{M}_1) \subseteq \mathscr{M}_2$, i.e., $\mathscr{F}_{\mathrm{HG}}$ preserves injectivity of morphisms, and $\mathscr{F}_{\mathrm{HG}}$ preserves pushouts along $\mathscr{M}$-morphisms (see [MEE13]).

Now we apply the translation and creation of (direct) transformations (see Theorem 1) to the $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathscr{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathscr{M}_2)$ leading to our new main technical result.

[3] where $E^{G'}_s \cong E^{G'}_{n2e}$ and $E^{G'}_t \cong E^{G'}_{e2n}$.

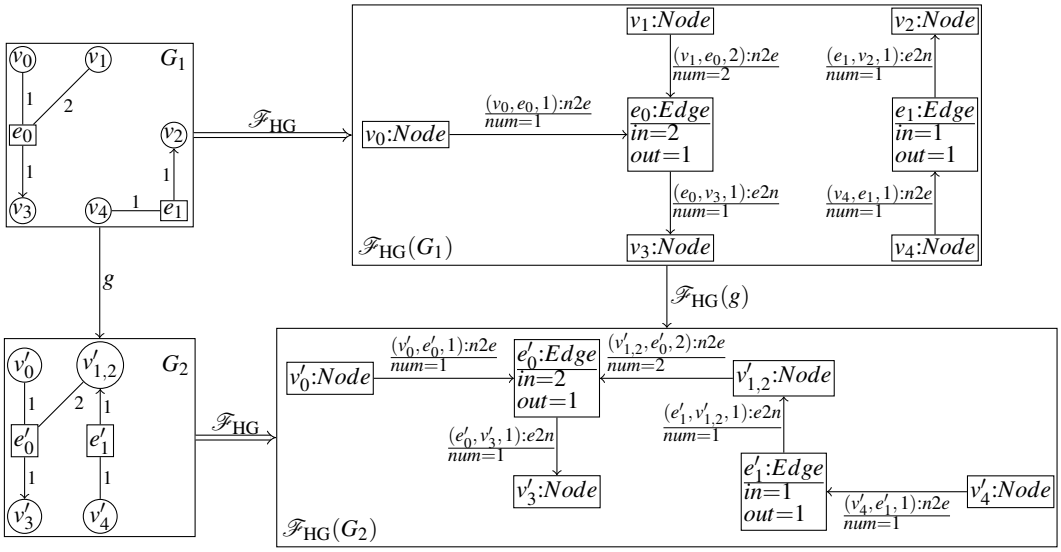


Figure 3: Applying functor $\mathscr{F}_{\mathrm{HG}}$ to two hypergraphs and morphism between them

Theorem 2 (Translation and Creation of Transformations between Hypergraphs and Typed Attributed Graphs)
*The $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}}$: (**HyperGraphs**, $\mathscr{M}_1$) $\to$ (**AGraphs$_{\mathbf{HGTG}}$**, $\mathscr{M}_2$) translates and creates direct transformations and transformations.*

Proof Idea. According to Theorem 1 we have to show that $\mathscr{F}_{\mathrm{HG}}$ creates morphisms and preserves initial pushouts.

1. ($\mathscr{F}_{\mathrm{HG}}$ **creates morphisms):** Given a typed attributed graph morphism $f' : \mathscr{F}_{\mathrm{HG}}(G_1) \to \mathscr{F}_{\mathrm{HG}}(G_2)$, there is a unique hypergraph morphism $f : G_1 \to G_2$ with $\mathscr{F}_{\mathrm{HG}}(f) = f'$ defined by $f = (f_V, f_E)$ with $f_V(v) = f'_{V_G}(v)$ for $v \in V_{G_1} \subseteq V_G^{G_1}$ and $f_E(e) = f'_{V_G}(e)$ for $e \in E_{G_1} \subseteq V_G^{G_1}$, where $V_G^{G_1} = V_{G_1} \uplus E_{G_1}$ is the V_G-component of $\mathscr{F}_{\mathrm{HG}}(G_1)$. From the morphism property of f' we can show that f is a hypergraph morphism with $\mathscr{F}_{\mathrm{HG}}(f) = f'$ and $\mathscr{F}_{\mathrm{HG}}(f) = \mathscr{F}_{\mathrm{HG}}(g)$ implies $f = g$ and hence uniqueness. The proof is based on the lemma given in [MEE13] showing that each typed attributed graph morphism $f' : \mathscr{F}_{\mathrm{HG}}(G_1) \to \mathscr{F}_{\mathrm{HG}}(G_2)$ is uniquely determined by its V_G-component $f'_{V_G} : V_G^{G_1} \to V_G^{G_2}$.
2. ($\mathscr{F}_{\mathrm{HG}}$ **preserves initial pushouts):** Preservation of initial pushouts means that (**HyperGraphs**, $\mathscr{M}_1$) has initial pushouts, which become also initial pushouts in (**AGraphs$_{\mathbf{HGTG}}$**, $\mathscr{M}_2$) as defined in [EEPT06] after application of $\mathscr{F}_{\mathrm{HG}}$. The construction of initial pushouts in (**HyperGraphs**, $\mathscr{M}_1$) and their preservation is shown in [MEE13]. □

4 $\mathscr{F}$-Transfer of Local Confluence

In this section, we review under which conditions local confluence can be translated by $\mathscr{M}$-functor $\mathscr{F}$: ($\mathbf{C_1}, \mathscr{M}_1$) $\to$ ($\mathbf{C_2}, \mathscr{M}_2$) from one transformation system $AS_1 = (\mathbf{C_1}, \mathscr{M}_1, P)$ to another

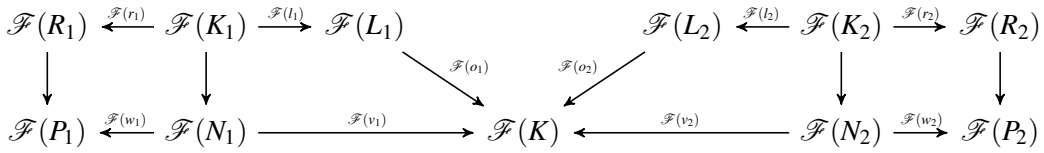
one $AS_2 = (\mathbf{C_2}, \mathscr{M}_2, \mathscr{F}(P))$ with translated productions $\mathscr{F}(P)$ and, vice versa, under which conditions local confluence of AS_1 can be created by $\mathscr{F}$ from local confluence of AS_2 (see [MEE12]). In this case, we speak of *$\mathscr{F}$-transfer of local confluence*.

According to [EEPT06], an $\mathscr{M}$-adhesive transformation system $(\mathbf{C}, \mathscr{M}, P)$ is locally confluent, if for all direct transformations $G \Rightarrow H_1$ and $G \Rightarrow H_2$ there is an object X together with transformations $H_1 \overset{*}{\Rightarrow} X$ and $H_2 \overset{*}{\Rightarrow} X$. In the case of confluence this property is required for transformations $G \overset{*}{\Rightarrow} H_1$ and $G \overset{*}{\Rightarrow} H_2$.

In [MEE12] it is shown under the assumptions of Theorem 1 that AS_1 is locally confluent for all transformation spans $H_1 \overset{\rho_1, m_1}{\Longleftarrow} G \overset{\rho_2, m_2}{\Longrightarrow} H_2$ iff AS_2 is locally confluent for all translated transformation spans $\mathscr{F}(H_1) \overset{\mathscr{F}(\rho_1), \mathscr{F}(m_1)}{\Longleftarrow} \mathscr{F}(G) \overset{\mathscr{F}(\rho_2), \mathscr{F}(m_2)}{\Longrightarrow} \mathscr{F}(H_2)$.

A well-known approach for the verification of local confluence is the analysis of critical pairs. A critical pair $P_1 \overset{\rho_1, o_1}{\Longleftarrow} K \overset{\rho_2, o_2}{\Longrightarrow} P_2$ is a pair of parallel dependent transformations with a minimal overlapping K of the left-hand sides of the rules.

Definition 5 ($\mathscr{F}$-Reachable Critical Pair)
Given an $\mathscr{M}$-functor $\mathscr{F} : (\mathbf{C_1}, \mathscr{M}_1) \to (\mathbf{C_2}, \mathscr{M}_2)$. An $\mathscr{F}$-reachable critical pair of productions $\mathscr{F}(\rho_1)$ and $\mathscr{F}(\rho_2)$ is a critical pair in AS_2 of the form

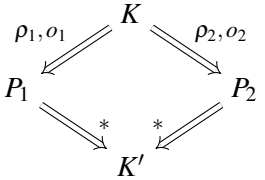


where all morphisms of type $\mathscr{F}(A) \to \mathscr{F}(B)$ are of the form $\mathscr{F}(f)$ for some morphism $f : A \to B$.

Note that for determining $\mathscr{F}$-reachability of a critical pair, it is sufficient to ensure that the overlapping of $\mathscr{F}(L_1)$ and $\mathscr{F}(L_2)$ is an $\mathscr{F}$-image [MEE12].

For Theorem 3 below we require that $\mathscr{F} : (\mathbf{C_1}, \mathscr{M}_1) \to (\mathbf{C_2}, \mathscr{M}_2)$ is *compatible with pair factorisation*. This means, on the one hand, that $(\mathbf{C_i}, \mathscr{M}_i)$ has pair factorisation based on $\mathscr{E}_i - \mathscr{M}_i$-factorisation for $i \in \{1,2\}$. For $(\mathbf{C_1}, \mathscr{M}_1)$ this means that each morphism pair $(f_1 : L_1 \to G, f_2 : L_2 \to G)$ with common codomain can be decomposed uniquely up to isomorphism as $(f_1 = m \circ e_1, f_2 = m \circ e_2)$ with a pair (e_1, e_2) of jointly epimorphic morphisms and $m \in \mathscr{M}_1$. On the other hand, it means that $\mathscr{F}$ preserves pair factorisation, i.e., for each pair factorisation $(f_1 = m \circ e_1, f_2 = m \circ e_2)$ in $(\mathbf{C_1}, \mathscr{M}_1)$ also $(\mathscr{F}(f_1) = \mathscr{F}(m) \circ \mathscr{F}(e_1), \mathscr{F}(f_2) = \mathscr{F}(m) \circ \mathscr{F}(e_2))$ is a pair factorisation in $(\mathbf{C_2}, \mathscr{M}_2)$.

Furthermore, we use the *Local Confluence Theorem* [EEPT06] to analyse whether a given $\mathscr{M}$-adhesive transformation system is locally confluent. This is the case, if all critical pairs $P_1 \overset{\rho_1, o_1}{\Longleftarrow} K \overset{\rho_2, o_2}{\Longrightarrow} P_2$ of the given transformation system are strictly confluent. Strictness means intuitively that the largest substructure of K that is preserved by the critical pair is also preserved by the merging transformation steps $P_1 \overset{*}{\Rightarrow} K'$ and $P_2 \overset{*}{\Rightarrow} K'$ (see the diagram to the right).



The following Theorem 3, with the proof in [MEE12], shows that AS_1 is locally confluent if all $\mathscr{F}$-reachable critical pairs in AS_2 are strictly confluent. This is important if in AS_2 critical pairs of typed attributed graph transformation systems have to be considered, because they can be detected automatically using the tool AGG.

Theorem 3 (Creation of Local Confluence Based on $\mathscr{F}$-Reachable Critical Pairs)
Given $\mathscr{M}$-adhesive transformation systems $AS_1 = (\mathbf{C_1}, \mathscr{M}_1, P)$, $AS_2 = (\mathbf{C_2}, \mathscr{M}_2, \mathscr{F}(P))$ and an $\mathscr{M}$-functor $\mathscr{F} : (\mathbf{C_1}, \mathscr{M}_1) \to (\mathbf{C_2}, \mathscr{M}_2)$ that creates (direct) transformations and morphisms (see Theorem 1 in Section 2) and is compatible with pair factorisation in the sense as discussed before. Then, AS_1 is locally confluent for all transformation spans $H_1 \overset{\rho_1,m_1}{\Longleftarrow} G \overset{\rho_2,m_2}{\Longrightarrow} H_2$ if all $\mathscr{F}$-reachable critical pairs of $\mathscr{F}(\rho_1)$ and $\mathscr{F}(\rho_2)$ in AS_2 are strictly confluent.

Now we apply the results concerning the creation of local confluence based on $\mathscr{F}$-reachable critical pairs to the concrete $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathscr{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathscr{M}_2)$. This is our main conceptual result, allowing us to use AGG for the analysis of hypergraph transformation systems.

Theorem 4 (Local Confluence of Hypergraph Transformation Systems)
Consider the $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathscr{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathscr{M}_2)$ from Definition 4 in Section 3. A hypergraph transformation system is locally confluent for all transformation spans $H_1 \overset{\rho_1,m_1}{\Longleftarrow} G \overset{\rho_2,m_2}{\Longrightarrow} H_2$ if all $\mathscr{F}_{\mathrm{HG}}$-reachable critical pairs of $\mathscr{F}_{\mathrm{HG}}(\rho_1)$ and $\mathscr{F}_{\mathrm{HG}}(\rho_2)$ are strictly confluent.

Proof Idea. In Theorem 2 (see Section 3), we have shown that $\mathscr{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathscr{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathscr{M}_2)$ is an $\mathscr{M}$-functor, which creates (direct) transformations and morphisms. Moreover, $\mathscr{F}_{\mathrm{HG}}$ is compatible with pair factorisation using the $\mathscr{E}$-$\mathscr{M}$-factorisations $(\mathscr{E}_1, \mathscr{M}_1)$ in $(\mathbf{HyperGraphs}, \mathscr{M}_1)$ and $(\mathscr{E}_2, \mathscr{M}_2)$ in $(\mathbf{AGraphs_{HGTG}}, \mathscr{M}_2)$, where $\mathscr{E}_1$ and $\mathscr{E}_2$ are the classes of surjective morphisms. In fact, $\mathscr{F}_{\mathrm{HG}}$ preserves coproducts and we have $\mathscr{F}_{\mathrm{HG}}(\mathscr{E}_1) \subseteq \mathscr{E}_2$ such that we obtain compatibility of $\mathscr{F}_{\mathrm{HG}}$ with pair factorisation according to [MEE13]. Altogether, this allows us to apply Theorem 3 with $\mathscr{F} = \mathscr{F}_{\mathrm{HG}}$. □

Since $\mathbf{AGraphs_{HGTG}}$ is the category of typed attributed graph transformation systems, we can use the tool AGG for critical pair analysis, while it is sufficient for our result to consider only $\mathscr{F}_{\mathrm{HG}}$-reachable critical pairs (see Definition 5 in this section).

5 Analysis of Hypergraph Transformation Systems based on AGG

We consider a simple distributed system with mobility, inspired by [BCK02], with servers connected by channels, and processes moving through the network and running on the servers.

Note that for this example, we use hypergraphs extended by a labelling function for hyperedges. Objects in this slightly extended category have the form: $G = (V_G, E_G, s_G, t_G, l_G)$ with the labelling function $l_G : E_G \to A$, where A is some alphabet. The corresponding $\mathscr{M}$-functor additionally translates the hyperedge labels into String attributes of the corresponding hyperedge node representation. All properties shown in Section 3 and Section 4 do also hold for the $\mathscr{M}$-adhesive

category of labelled hypergraphs and the extended $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}}$.[4]

In our distributed system model with mobility, servers, connections and processes are represented as labelled hyperedges. The meaning of the hyperedge labels is as follows: P denotes a process before it is executed, S stands for server, and C for connection. A running process is represented by label R. Note that, on the one hand, we simplify the network model in [BCK02] by disregarding firewalls and secure servers; on the other hand, we allow for connections between three servers modelled by hyperedges with three tentacles, and we distinguish between travelling processes P and running processes R.

The hypergraph in Figure 4 models a network with four servers, different kinds of connections between them, and two processes. Nodes are depicted as black bullets, while hyperedges are represented by labelled rectangles.

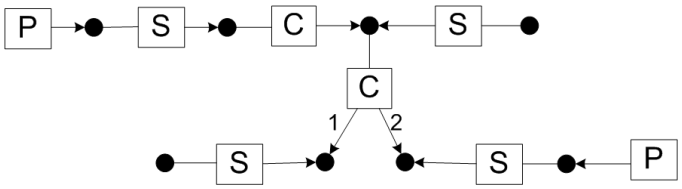


Figure 4: Hypergraph defining a network with distributed processes

The behaviour of the system is modelled by the hypergraph transformation rules in Figure 5. Rules enterServer [leaveServer] allow a process to enter [leave] a server location. Both rules are inverse to each other (indicated by the double arrow). Rules crossC [backC] model the travelling of a process via a connection. We have different rules for process travelling, depending on the kind of connection hyperedge that is crossed. When a process finally has found a suitable server, it switches into the state *running* by applying the rule runP. A process that has finished its execution is removed from the system by the rule removeR.

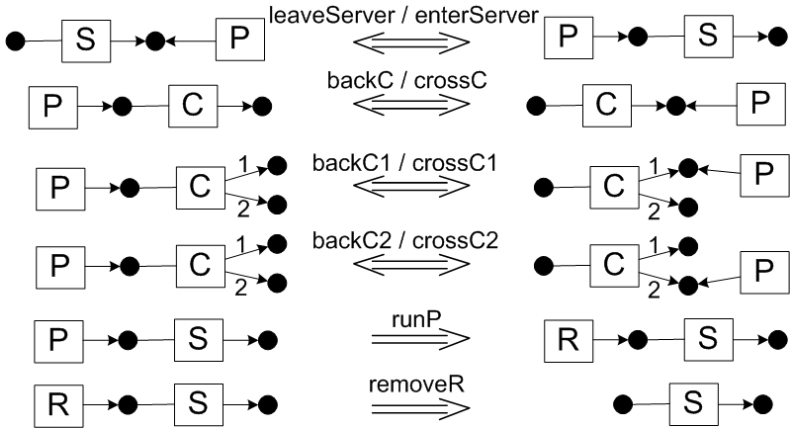


Figure 5: Hypergraph transformation rules modelling the behaviour of mobile processes

[4] Note that this holds also for the variants of hypergraphs with labelled nodes and/or labelled hyperedges.

Applying the $\mathscr{M}$-functor $\mathscr{F}_{\mathrm{HG}}$ from Definition 4 to this hypergraph transformation system results in a typed attributed graph transformation system that can be statically analysed using AGG. Figure 6 shows an $\mathscr{F}_{\mathrm{HG}}$-reachable critical pair for the application of rules leaveServer and crossC (depicted in the left part) when their left-hand sides overlap as indicated in the overlapping graph in the right part. The conflict (a delete-use conflict) is obviously caused when a process can "choose" either to leave a server location or to cross a connection channel in the network.

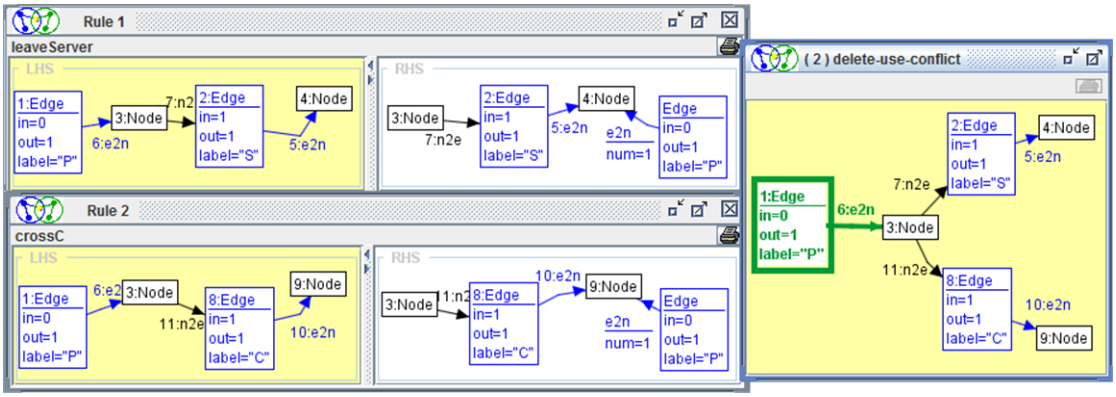


Figure 6: A critical pair detected by AGG for the rule pair (leaveServer, crossC)

This critical pair is strictly confluent: after applying either rule, we can reverse the effect by applying the corresponding inverse rule and hence have at least one graph which can be derived to join the different results. We can even conclude that the whole transformation system is locally confluent by applying Theorem 4, showing strict confluence of all $\mathscr{F}_{\mathrm{HG}}$-reachable critical pairs.[5] Obviously, rules for process travelling can be executed in any order; a step modelling forward travelling and a step modelling the execution of a process can be joined again by performing backward travelling and then executing the process.

6 Related Work and Conclusion

In our previous paper [MEE11] we have developed a general framework to establish a formal relationship between different $\mathscr{M}$-adhesive transformation systems, showing under which conditions transformations can be translated and created between different $\mathscr{M}$-adhesive transformation systems. This result is based on suitable properties of $\mathscr{M}$-functors between the corresponding $\mathscr{M}$-adhesive categories. In this paper, we construct an $\mathscr{M}$-functor from hypergraphs to typed attributed graphs. We show in our main technical result in Theorem 2 (with non-trivial proof in [MEE13]) that the functor satisfies the required properties guaranteeing translation and creation of rule applications, as well as the transfer of local confluence. Moreover, also termination and functional behavior can be transferred according to [MEE12]. This provides us with a general framework to analyse hypergraph transformation systems and allows us by Theorem 4 to use

[5] Note that there are several non-$\mathscr{F}_{\mathrm{HG}}$-reachable critical pairs that do not have to be considered according to Theorem 4.

the critical pair analysis of the AGG-tool [AGG12] for typed attributed graphs to analyse confluence of hypergraph transformation systems. We demonstrate this by analysing a hypergraph transformation system modelling a distributed system with mobile processes.

A related approach to hypergraph analysis considers causal dependencies modelled by approximated unfolding [BCK02, BK02]. The thesis of D. Plump [Plu93] contains already theoretical results about confluence of hypergraph transformation systems, comprising a sufficient condition for local confluence based on critical pairs. But to the best of our knowledge, a tool supporting directly critical pair analysis of hypergraph transformation systems does not yet exist.

A suitable automated detection of $\mathscr{F}$-reachable critical pairs would be helpful to reduce the analysis effort, and is subject to future work. Furthermore, we will investigate how (nested) application conditions [HP05] can be handled in this framework in order to consider critical pairs and local confluence of $\mathscr{M}$-adhesive transformation systems with (nested) application conditions.

References

[AGG12] TFS-Group, TU Berlin. AGG. 2012. http://www.tfs.tu-berlin.de/agg.

[BCK02] P. Baldan, A. Corradini, B. König. Static Analysis of Distributed Systems with Mobility Specified by Graph Grammars—A Case Study. In Ehrig et al. (eds.), *Proc. of Int. Conf. on Integrated Design & Process Technology*. SDPS, 2002.

[BK02] P. Baldan, B. König. Approximating the behaviour of graph transformation systems. In *Int. Conf. on Graph Transformation*. Pp. 14–29. Springer, 2002. LNCS 2505.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theor. Comp. Science. Springer, 2006.

[EGH10] H. Ehrig, U. Golas, F. Hermann. Categorical Frameworks for Graph Transformation and HLR Systems based on the DPO Approach. *EATCS Bulletin* 102:111–121, 2010.

[HP05] A. Habel, K.-H. Pennemann. Nested constraints and application conditions for high-level structures. In Kreowski et al. (eds.), *Formal Methods in Software and Systems Modeling*. LNCS 3393, pp. 294–308. Springer, 2005.

[LS04] S. Lack, P. Sobociński. Adhesive Categories. In *Proc. FOSSACS' 04*. LNCS 2987, pp. 273–288. Springer, 2004.

[MEE11] M. Maximova, H. Ehrig, C. Ermel. Formal Relationship between Petri Net and Graph Transformation Systems based on Functors between $\mathscr{M}$-adhesive Categories. In *Proc. of 4th Workshop on Petri Nets and Graph Transformation Systems*. Volume 40. ECEASST, 2011. http://journal.ub.tu-berlin.de/index.php/eceasst/issue/archive.

[MEE12] M. Maximova, H. Ehrig, C. Ermel. Transfer of Local Confluence and Termination between Petri Net and Graph Transformation Systems Based on $\mathscr{M}$-Functors. In *Proc. of 5th Workshop on Petri Nets and Graph Transformation Systems*. Volume 51, pp. 1–12. ECEASST, 2012. http://journal.ub.tu-berlin.de/index.php/eceasst/issue/archive.

[MEE13] M. Maximova, H. Ehrig, C. Ermel. Analysis of Hypergraph Transformation Systems in AGG based on $\mathscr{M}$-Functors: Extended Version. Technical report 2013/02, TU Berlin, 2013. http://www.eecs.tu-berlin.de/menue/forschung/forschungsberichte/2013.

[Min00] M. Minas. Hypergraphs as a Uniform Diagram Representation Model. In *Proc. 6th Int. Workshop on Theory and Application of Graph Transformations (TAGT'98)*. LNCS 1764, pp. 281–295. Springer, 2000.

[Plu93] D. Plump. *Evaluation of functional expressions by hypergraph rewriting*. PhD thesis, Universität Bremen, Fachbereich Mathematik und Informatik, 1993.