Technische Universität Berlin

**Forschungsberichte
der Fakultät IV – Elektrotechnik und Informatik**

Analysis of Hypergraph Transformation Systems in AGG based on $\mathcal{M}$-Functors: Extended Version

Maria Maximova,
Hartmut Ehrig,
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

Analysis of Hypergraph Transformation Systems in AGG based on $\mathcal{M}$-Functors: Extended Version

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 $\mathcal{M}$-adhesive transformation systems based on $\mathcal{M}$-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 $\mathcal{M}$-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 $\mathcal{M}$-functors between different $\mathcal{M}$-adhesive transformation systems. We construct a functor between the $\mathcal{M}$-adhesive categories of hypergraphs and of typed attributed graphs, and show that our construction yields an $\mathcal{M}$-functor with suitable properties. We then use existing results for $\mathcal{M}$-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: $\mathcal{M}$-adhesive transformation system, graph transformation, hypergraph transformation, $\mathcal{M}$-adhesive category, $\mathcal{M}$-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 [11]. Other applications are distributed systems [1] and diagram representation [10].

Hypergraph transformation is related to algebraic graph transformation [3], where structural changes are modelled in the double-pushout (DPO) approach for the category of (typed, attributed) graphs, which has been generalised to $\mathcal{M}$-adhesive categories, relying on a class $\mathcal{M}$ 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 [3]. The well-established tool AGG [12] 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 [7], 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 [7], we established the new framework of $\mathcal{M}$-functors $\mathcal{F} : (\mathbf{C_1}, \mathcal{M}_1) \to (\mathbf{C_2}, \mathcal{M}_2)$ between $\mathcal{M}$-adhesive categories. This framework allows to translate transformations in $(\mathbf{C_1}, \mathcal{M}_1)$ into corresponding transformations in $(\mathbf{C_2}, \mathcal{M}_2)$ and, vice versa, to create transformations in $(\mathbf{C_1}, \mathcal{M}_1)$ from those in $(\mathbf{C_2}, \mathcal{M}_2)$.

Building on this previous work, we have extended this framework in [8] 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 $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}} : (\mathbf{C_1}, \mathcal{M}_1) \to (\mathbf{C_2}, \mathcal{M}_2)$, where $(\mathbf{C_1}, \mathcal{M}_1)$ are hypergraphs and $(\mathbf{C_2}, \mathcal{M}_2)$ typed attributed graphs. In our main results, we show that the functor $\mathcal{F}_{\mathrm{HG}}$ 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 [12] to analyse these properties for hypergraph transformations. In contrast to previous instantiations of $\mathcal{M}$-functors in [7], we do not have to restrict our functor to injective morphisms.

This technical report is an extended version of our paper [9]. It is structured as follows: Section 2 introduces the basic notions of $\mathcal{M}$-adhesive transformation systems and $\mathcal{M}$-functors to define a formal relationship between two different $\mathcal{M}$-adhesive categories. In Section 3, we construct the functor $\mathcal{F}_{\mathrm{HG}}$ between hypergraph and typed attributed graph transformation systems, and show that this functor satisfies the properties of $\mathcal{M}$-functors and some additional properties

that are required in the general theory. In Section 4, we study the $\mathcal{F}$-transfer of local confluence by analysing $\mathcal{F}$-reachable critical pairs and show that the $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}}$ 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 Appendix A.

2 $\mathcal{M}$-Adhesive Categories, Transformation Systems, $\mathcal{M}$-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 [7]. Our considerations are based on the framework of $\mathcal{M}$-adhesive categories. An $\mathcal{M}$-adhesive category [4], consists of a category $\mathbf{C}$ together with a class $\mathcal{M}$ of monomorphisms such that the following properties hold: $\mathbf{C}$ has pushouts (POs) and pullbacks (PBs) along $\mathcal{M}$-morphisms, $\mathcal{M}$ is closed under isomorphisms, composition, decomposition, POs and PBs, and POs along $\mathcal{M}$-morphisms are $\mathcal{M}$-VK-squares (see Figure 1), 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 $\mathcal{M}$. The VK-property means that the top face is a PO iff the front faces are PBs.
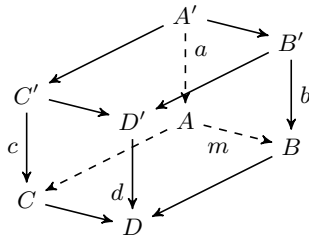

Fig. 1. $\mathcal{M}$-VK-square

The concept of $\mathcal{M}$-adhesive categories generalises that of adhesive [6], adhesive HLR, and weak adhesive HLR categories [3]. The categories of typed attributed graphs, hypergraphs and several categories of Petri nets are weak adhesive HLR (see [3]) and hence also $\mathcal{M}$-adhesive. A set of transformation rules in an $\mathcal{M}$-adhesive category constitutes an $\mathcal{M}$-adhesive transformation system [4].

Definition 1 ($\mathcal{M}$-Adhesive Transformation System).
Given an $\mathcal{M}$-adhesive category $(\mathbf{C}, \mathcal{M})$, an $\mathcal{M}$-adhesive transformation system

3

$AS = (\mathbf{C}, \mathcal{M}, P)$ *has a set P of productions of the form* $\rho = (L \xleftarrow{l} K \xrightarrow{r} R)$ *with* $l, r \in \mathcal{M}$. *A direct transformation* $G \stackrel{\rho,m}{\Longrightarrow} H$ *via ρ and match m consists of two pushouts according to the DPO approach [3].*

We use the notion of an $\mathcal{M}$-functor [7] to define a formal relationship between two different $\mathcal{M}$-adhesive transformation systems.

Definition 2 ($\mathcal{M}$-Functor).
A functor $\mathcal{F} : (\mathbf{C_1}, \mathcal{M}_1) \to (\mathbf{C_2}, \mathcal{M}_2)$ 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.

Given an $\mathcal{M}$-adhesive transformation system $AS_1 = (\mathbf{C_1}, \mathcal{M}_1, P_1)$, we want to translate transformations from AS_1 to $AS_2 = (\mathbf{C_2}, \mathcal{M}_2, P_2)$ with translated productions $P_2 = \mathcal{F}(P_1)$ 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 [7].
By definition, each $\mathcal{M}$-functor $\mathcal{F} : (\mathbf{C_1}, \mathcal{M}_1) \to (\mathbf{C_2}, \mathcal{M}_2)$ translates each production $\rho = (L \xleftarrow{l} K \xrightarrow{r} R)$ in P_1 with $l, r \in \mathcal{M}_1$ into $\mathcal{F}(\rho) = (\mathcal{F}(L) \xleftarrow{\mathcal{F}(l)} \mathcal{F}(K) \xrightarrow{\mathcal{F}(r)} \mathcal{F}(R))$ in $P_2 = \mathcal{F}(P_1)$ with $\mathcal{F}(l), \mathcal{F}(r) \in \mathcal{M}_2$ and each direct transformation $G \stackrel{\rho,m}{\Longrightarrow} H$ in AS_1 given by DPO $(1) + (2)$ into a direct transformation $\mathcal{F}(G) \stackrel{\mathcal{F}(\rho),\mathcal{F}(m)}{\Longrightarrow} \mathcal{F}(H)$ in AS_2 given by DPO $(3) + (4)$.

$$
\begin{array}{ccccc}
L & \xleftarrow{l} & K & \xrightarrow{r} & R \\
m\downarrow & (1) & \downarrow & (2) & \downarrow \\
G & \longleftarrow & D & \longrightarrow & H
\end{array}
\qquad \Longrightarrow \qquad
\begin{array}{ccccc}
\mathcal{F}(L) & \xleftarrow{\mathcal{F}(l)} & \mathcal{F}(K) & \xrightarrow{\mathcal{F}(r)} & \mathcal{F}(R) \\
\mathcal{F}(m)\downarrow & (3) & \downarrow & (4) & \downarrow \\
\mathcal{F}(G) & \longleftarrow & \mathcal{F}(D) & \longrightarrow & \mathcal{F}(H)
\end{array}
$$

Vice versa, we say $\mathcal{F}$ creates direct transformations, if for each direct transformation $\mathcal{F}(G) \stackrel{\mathcal{F}(\rho),m'}{\Longrightarrow} H'$ in AS_2 there is a direct transformation $G \stackrel{\rho,m}{\Longrightarrow} H$ in AS_1 with $\mathcal{F}(m) = m'$ and $\mathcal{F}(H) \cong H'$ leading to $\mathcal{F}(G) \stackrel{\mathcal{F}(\rho),\mathcal{F}(m)}{\Longrightarrow} \mathcal{F}(H)$ 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 $G \stackrel{*}{\Rightarrow} H$.

Theorem 1 (Translation and Creation of Transformations).
Each $\mathcal{M}$-functor $\mathcal{F} : (\mathbf{C_1}, \mathcal{M}_1) \to (\mathbf{C_2}, \mathcal{M}_2)$ translates (direct) transformations. Vice versa, $\mathcal{F}$ creates (direct) transformations if we have the following two conditions:
 - *($\mathcal{F}$ **creates morphisms**): For all $m' : \mathcal{F}(L) \to \mathcal{F}(G)$ in $(\mathbf{C_2}, \mathcal{M}_2)$, there is exactly one morphism $m : L \to G$ with $\mathcal{F}(m) = m'$.*

4

– **(F preserves initial pushouts):** $(\mathbf{C_1}, \mathcal{M}_1)$ *has initial pushouts and for each initial pushout* (1) *over* $m : L \to G$, *also* (2) *is initial pushout over* $\mathcal{F}(m)$ *in* $(\mathbf{C_2}, \mathcal{M}_2)$.

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

The proof for Theorem 1 is given in [7]. Moreover, it is shown under the same assumptions that $\mathcal{F}$ 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 [3,7].

3 $\mathcal{M}$-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 [7] and [8]. For this purpose, we review on the one hand the category $(\mathbf{HyperGraphs}, \mathcal{M}_1)$ of hypergraphs with the class $\mathcal{M}_1$ of all injective hypergraph morphisms, which is shown to be $\mathcal{M}$-adhesive in [3]. On the other hand, we review the category of typed attributed graphs $(\mathbf{AGraphs_{ATG}}, \mathcal{M}_2)$ with the class $\mathcal{M}_2$ of all injective typed attributed graph morphisms, which is also shown to be $\mathcal{M}$-adhesive in [3], and we define a suitable attributed hypergraph type graph $ATG = HGTG$. Moreover, we construct a functor $\mathcal{F}_{\mathrm{HG}}$ between both categories and show that the general result from Theorem 1 is applicable to this functor.
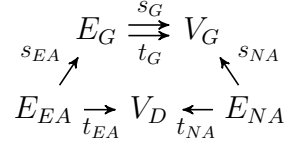
Definition 3 (Category HyperGraphs [3]).
A hypergraph G is defined as $G = (V_G, E_G, s_G, t_G)$, where V_G is a set of hypergraph nodes, E_G is a set of hyperedges and $s_G, t_G : E_G \to V_G^*$ are functions assigning the string $s_G(e)$ of source nodes resp. $t_G(e)$ of target nodes to each hyperedge 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 $x_1 \ldots x_n \mapsto f_V(x_1) \ldots f_V(x_n)$.

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

According to [3], the category $(\mathbf{HyperGraphs}, \mathcal{M}_1)$ of hypergraphs with a class $\mathcal{M}_1$ of all injective morphisms is $\mathcal{M}$-adhesive, where pushouts are constructed componentwise.

5

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

$$
\begin{array}{ccc}
 & E_G \underset{t_G}{\overset{s_G}{\rightrightarrows}} V_G & \\
s_{EA} \nearrow & & \nwarrow s_{NA} \\
E_{EA} \underset{t_{EA}}{\rightarrow} V_D \underset{t_{NA}}{\leftarrow} E_{NA} &
\end{array}
$$

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 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 **AGraphs$_{\mathbf{ATG}}$**, where attributed graphs are typed over an attributed type graph ATG [3]. 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.

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

According to [3], the category (**AGraphs$_{\mathbf{ATG}}$**, $\mathcal{M}$) is $\mathcal{M}$-adhesive for each type graph ATG, where $\mathcal{M}$-morphisms are injective with isomorphic data type part.

---

[1] Node and edge attributes are depicted in compact notation as node/edge inscriptions together with their data type.
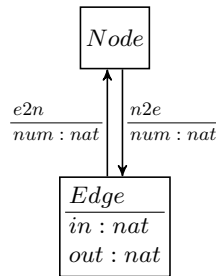
Fig. 2. Attributed type graph $HGTG$

Hence also the special case of $(\mathbf{AGraphs_{ATG}}, \mathcal{M})$ with $ATG = HGTG$ is $\mathcal{M}$-adhesive. The subcategory $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ with identical algebra homomorphism as considered above is also $\mathcal{M}$-adhesive for the subclass $\mathcal{M} = \mathcal{M}_2$ of all injective typed attributed graph morphisms.

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

Definition 4 ($\mathcal{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathcal{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$)**.**
Consider a hypergraph $G = (V_G, E_G, s_G, t_G)$. *We define the object* $\mathcal{F}_{\mathrm{HG}}(G) = ((G', NAT), type)^2$ *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:*

$$V_G^{G'} = V_G \uplus E_G \text{ (graph nodes)}$$
$$E_G^{G'} = E_{n2e}^{G'} \uplus E_{e2n}^{G'} \text{ (graph edges) with}$$
$$\quad E_{n2e}^{G'} = \{(v, e, n) \in V_G \times E_G \times \mathbb{N} \mid s_G^n(e) = v\},$$
$$\quad E_{e2n}^{G'} = \{(e, v, n) \in E_G \times V_G \times \mathbb{N} \mid t_G^n(e) = v\},$$
$$\quad \text{where } s_G^n(e) \text{ is the n-th node in the string } s_G(e) \text{ and similar for } t_G^n(e),$$
$$E_{NA}^{G'} = E_{in}^{G'} \uplus E_{out}^{G'} \text{ (node attribute edges) with}$$
$$\quad E_{in}^{G'} = \{(e, n, in) \mid (e, n) \in E_G \times \mathbb{N} \wedge |s_G(e)| = n\},$$
$$\quad E_{out}^{G'} = \{(e, n, out) \mid (e, n) \in E_G \times \mathbb{N} \wedge |t_G(e)| = n\}, \text{ where } |w| \text{ is the}$$
$$\quad \text{length of the string } w,$$
$$E_{EA}^{G'} = E_s^{G'} \uplus E_t^{G'} \text{ (edge attribute edges) with}$$
$$\quad E_s^{G'} = \{(n, v, e) \in \mathbb{N} \times V_G \times E_G \mid s_G^n(e) = v\}^3,$$
$$\quad E_t^{G'} = \{(n, e, v) \in \mathbb{N} \times E_G \times V_G \mid t_G^n(e) = v\}^3,$$
$$\text{(and the corresponding source and target functions:)}$$
$$s_G^{G'}, t_G^{G'} : E_G^{G'} \to V_G^{G'} \text{ defined by } s_G^{G'}(x, y, n) = x, \ t_G^{G'}(x, y, n) = y,$$
$$s_{NA}^{G'} : E_{NA}^{G'} \to V_G^{G'} \text{ defined by } s_{NA}^{G'}(e, n, x) = e$$
$$t_{NA}^{G'} : E_{NA}^{G'} \to \mathbb{N} \text{ defined by } t_{NA}^{G'}(e, n, x) = n,$$
$$s_{EA}^{G'} : E_{EA}^{G'} \to E_G^{G'} \text{ defined by } s_{EA}^{G'}(n, x, y) = (x, y, n),$$
$$t_{EA}^{G'} : E_{EA}^{G'} \to \mathbb{N} \text{ defined by } t_{EA}^{G'}(n, x, y) = n.$$

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* $type^{G'}$:

---
[2] In the following, we also use the short notation $\mathcal{F}_{\mathrm{HG}}(G) = G'$.

$G' \to HGTG$ 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., $type_{V_G} : V_G^{G'} \to V_G^{HGTG}$ 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 $\mathcal{F}_{HG}(f) : \mathcal{F}_{HG}(G_1) \to \mathcal{F}_{HG}(G_2)$, where in short notation $\mathcal{F}_{HG}(G_i) = (V_G^{G_i}, \mathbb{N}, E_G^{G_i}, E_{NA}^{G_i}, E_{EA}^{G_i}, (s_j^{G_i}, t_j^{G_i})_{j \in \{G, NA, EA\}})$ with $i \in \{1, 2\}$ by $\mathcal{F}_{HG}(f) = f' = (f'_{V_G}, f'_{V_D} = id_{\mathbb{N}}, f'_{E_G}, f'_{E_{NA}}, f'_{E_{EA}})$ with

$$f'_{V_G} : V_G^{G_1} \to V_G^{G_2} \quad with \quad V_G^{G_i} = V_{G_i} \uplus E_{G_i} \quad for \quad i \in \{1, 2\} \quad by \quad f'_{V_G} = f_V \uplus f_E$$

$$f'_{E_G} : E_G^{G_1} \to E_G^{G_2} \quad with \quad E_G^{G_i} = E_{n2e}^{G_i} \uplus E_{e2n}^{G_i} \quad for \quad i \in \{1, 2\} \quad by$$

$$f'_{E_G}(v, e, n) = (f_V(v), f_E(e), n) \quad for \quad (v, e, n) \in E_{n2e}^{G_1}$$

$$f'_{E_G}(e, v, n) = (f_E(e), f_V(v), n) \quad for \quad (e, v, n) \in E_{e2n}^{G_1}$$

$$f'_{E_{NA}} : E_{NA}^{G_1} \to E_{NA}^{G_2} \quad with \quad E_{NA}^{G_i} = E_{in}^{G_i} \uplus E_{out}^{G_i} \quad for \quad i \in \{1, 2\} \quad by$$

$$f'_{E_{NA}}(e, n, x) = (f_E(e), n, x) \quad for \quad (e, n, x) \in E_{in}^{G_1} \uplus E_{out}^{G_1} \wedge x \in \{in, out\}$$

$$f'_{E_{EA}} : E_{EA}^{G_1} \to E_{EA}^{G_2} \quad with \quad E_{EA}^{G_i} = E_s^{G_i} \uplus E_t^{G_i} \quad for \quad i \in \{1, 2\} \quad similar^3 \ to$$

$$f'_{E_G} : E_G^{G_1} \to E_G^{G_2}$$

An example for using the functor $\mathcal{F}_{HG}$ 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 $\mathcal{F}_{HG}$ defined above is a well-defined $\mathcal{M}$-functor in the sense of Definition 4. This includes that the components of $\mathcal{F}_{HG}(f)$ are well-defined w.r.t. their codomain and that they are compatible with source and target functions as well as typing morphisms. $\mathcal{F}_{HG}$ is a functor, because $\mathcal{F}_{HG}$ preserves identities and composition. Moreover, $\mathcal{F}_{HG}$ is an $\mathcal{M}$-functor, because we have $\mathcal{F}_{HG}(\mathcal{M}_1) \subseteq \mathcal{M}_2$, i.e., $\mathcal{F}_{HG}$ preserves injectivity of morphisms, and $\mathcal{F}_{HG}$ preserves pushouts along $\mathcal{M}$-morphisms (see Lemma 1 in Appendix A).

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

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

---

[3] where $E_s^{G'} \cong E_{n2e}^{G'}$ and $E_t^{G'} \cong E_{e2n}^{G'}$.
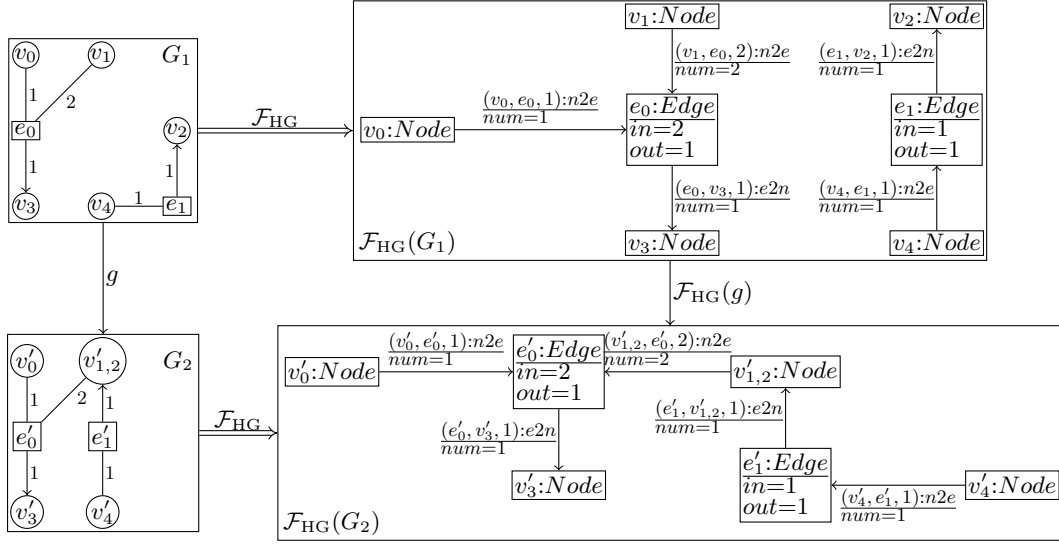
Fig. 3. Applying functor $\mathcal{F}_{\text{HG}}$ to two hypergraphs and morphism between them

Proof Idea.

According to Theorem 1 we have to show that $\mathcal{F}_{\text{HG}}$ creates morphisms and preserves initial pushouts.

1. ($\mathcal{F}_{\text{HG}}$ **creates morphisms**): Given a typed attributed graph morphism $f' : \mathcal{F}_{\text{HG}}(G_1) \to \mathcal{F}_{\text{HG}}(G_2)$, there is a unique hypergraph morphism $f : G_1 \to G_2$ with $\mathcal{F}_{\text{HG}}(f) = f'$ 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 $\mathcal{F}_{\text{HG}}(G_1)$. From the morphism property of f' we can show that f is a hypergraph morphism with $\mathcal{F}_{\text{HG}}(f) = f'$ and $\mathcal{F}_{\text{HG}}(f) = \mathcal{F}_{\text{HG}}(g)$ implies $f = g$ and hence uniqueness. The proof is based on Lemma 2 in Appendix A showing that each typed attributed graph morphism $f' : \mathcal{F}_{\text{HG}}(G_1) \to \mathcal{F}_{\text{HG}}(G_2)$ is uniquely determined by its V_G-component $f'_{V_G} : V_G^{G_1} \to V_G^{G_2}$.

2. ($\mathcal{F}_{\text{HG}}$ **preserves initial pushouts**): Preservation of initial pushouts means that (**Hyper-Graphs**, $\mathcal{M}_1$) has initial pushouts, which become also initial pushouts in (**AGraphs$_{\text{HGTG}}$**, $\mathcal{M}_2$) as defined in [3] after application of $\mathcal{F}_{\text{HG}}$. The construction of initial pushouts in (**HyperGraphs**, $\mathcal{M}_1$) and their preservation is shown in Lemma 3 in Appendix A.

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

In this section, we review under which conditions local confluence can be translated by $\mathcal{M}$-functor $\mathcal{F} : (\mathbf{C_1}, \mathcal{M}_1) \to (\mathbf{C_2}, \mathcal{M}_2)$ from one transformation system $AS_1 = (\mathbf{C_1}, \mathcal{M}_1, P)$ to another one $AS_2 = (\mathbf{C_2}, \mathcal{M}_2, \mathcal{F}(P))$ with translated productions $\mathcal{F}(P)$ and, vice versa, under which conditions local confluence of AS_1 can be created by $\mathcal{F}$ from local confluence of AS_2 (see [8]). In this case, we speak of $\mathcal{F}$-transfer of local confluence.

According to [3], an $\mathcal{M}$-adhesive transformation system $(\mathbf{C}, \mathcal{M}, P)$ 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 [8] 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 $\mathcal{F}(H_1) \overset{\mathcal{F}(\rho_1), \mathcal{F}(m_1)}{\Longleftarrow} \mathcal{F}(G) \overset{\mathcal{F}(\rho_2), \mathcal{F}(m_2)}{\Longrightarrow} \mathcal{F}(H_2)$.

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

$$\mathcal{F}(R_1) \overset{\mathcal{F}(r_1)}{\leftarrow} \mathcal{F}(K_1) \overset{\mathcal{F}(l_1)}{\rightarrow} \mathcal{F}(L_1) \qquad\qquad \mathcal{F}(L_2) \overset{\mathcal{F}(l_2)}{\leftarrow} \mathcal{F}(K_2) \overset{\mathcal{F}(r_2)}{\rightarrow} \mathcal{F}(R_2)$$

$$\mathcal{F}(P_1) \overset{\mathcal{F}(w_1)}{\leftarrow} \mathcal{F}(N_1) \overset{\mathcal{F}(v_1)}{\longrightarrow} \mathcal{F}(K) \overset{\mathcal{F}(v_2)}{\longleftarrow} \mathcal{F}(N_2) \overset{\mathcal{F}(w_2)}{\rightarrow} \mathcal{F}(P_2)$$

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

Note that for determining $\mathcal{F}$-reachability of a critical pair, it is sufficient to ensure that the overlapping of $\mathcal{F}(L_1)$ and $\mathcal{F}(L_2)$ is an $\mathcal{F}$-image [8].

For Theorem 3 below we require that $\mathcal{F} : (\mathbf{C_1}, \mathcal{M}_1) \to (\mathbf{C_2}, \mathcal{M}_2)$ is *compatible with pair factorisation*. This means, on the one hand, that $(\mathbf{C_i}, \mathcal{M}_i)$ has pair factorisation based on $\mathcal{E}_i - \mathcal{M}_i$-factorisation for $i \in \{1, 2\}$. For $(\mathbf{C_1}, \mathcal{M}_1)$ 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 $m \in \mathcal{M}_1$. On the other hand, it means that $\mathcal{F}$ preserves pair factorisation, i.e., for each pair factorisation $(f_1 = m \circ e_1, f_2 = m \circ e_2)$ in $(\mathbf{C_1}, \mathcal{M}_1)$ also $(\mathcal{F}(f_1) = \mathcal{F}(m) \circ \mathcal{F}(e_1), \mathcal{F}(f_2) = \mathcal{F}(m) \circ \mathcal{F}(e_2))$ is a pair factorisation in $(\mathbf{C_2}, \mathcal{M}_2)$.

Furthermore, we use the *Local Confluence Theorem* [3] to analyse whether a given $\mathcal{M}$-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).
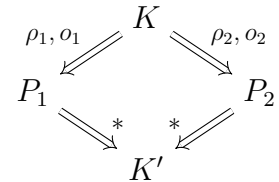
The following Theorem 3, with the proof in [8], shows that AS_1 is locally confluent if all $\mathcal{F}$-reachable critical pairs in AS_2 are strictly confluent. This is important if in AS_2 critical pairs of typed attributed graph transformation systems have to be considered, because they can be detected automatically using the tool AGG.

Theorem 3 (Creation of Local Confluence Based on $\mathcal{F}$-Reachable Critical Pairs).
Given $\mathcal{M}$-adhesive transformation systems $AS_1 = (\mathbf{C_1}, \mathcal{M}_1, P)$, $AS_2 = (\mathbf{C_2}, \mathcal{M}_2, \mathcal{F}(P))$ and an $\mathcal{M}$-functor $\mathcal{F} : (\mathbf{C_1}, \mathcal{M}_1) \to (\mathbf{C_2}, \mathcal{M}_2)$ 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 $\mathcal{F}$-reachable critical pairs of $\mathcal{F}(\rho_1)$ and $\mathcal{F}(\rho_2)$ in AS_2 are strictly confluent.

Now we apply the results concerning the creation of local confluence based on $\mathcal{F}$-reachable critical pairs to the concrete $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathcal{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$. 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 with Analysis using AGG).
Consider the $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathcal{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ 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 $\mathcal{F}_{\mathrm{HG}}$-reachable critical pairs of $\mathcal{F}_{\mathrm{HG}}(\rho_1)$ and $\mathcal{F}_{\mathrm{HG}}(\rho_2)$ are strictly confluent.

Proof Idea.
In Theorem 2 (see Section 3), we have shown that $\mathcal{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathcal{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ is an $\mathcal{M}$-functor, which creates (direct) transformations and morphisms. Moreover, $\mathcal{F}_{\mathrm{HG}}$ is compatible with pair factorisation using the $\mathcal{E}$-$\mathcal{M}$-factorisations $(\mathcal{E}_1, \mathcal{M}_1)$ in $(\mathbf{HyperGraphs}, \mathcal{M}_1)$ and $(\mathcal{E}_2, \mathcal{M}_2)$ in $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$, where $\mathcal{E}_1$ and $\mathcal{E}_2$ are the classes of surjective morphisms. In fact, $\mathcal{F}_{\mathrm{HG}}$ preserves coproducts and we have $\mathcal{F}_{\mathrm{HG}}(\mathcal{E}_1) \subseteq \mathcal{E}_2$ such that we obtain compatibility of $\mathcal{F}_{\mathrm{HG}}$ with pair factorisation according to Lemma 4 in Appendix A. Altogether, this allows us to apply Theorem 3 with $\mathcal{F} = \mathcal{F}_{\mathrm{HG}}$.

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 $\mathcal{F}_{\mathrm{HG}}$-reachable critical pairs (see Definition 5 in this section).

11

5 Analysis of Hypergraph Transformation Systems based on AGG

We consider a simple distributed system with mobility, inspired by [1], 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 $\mathcal{M}$-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 $\mathcal{M}$-adhesive category of labelled hypergraphs and the extended $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}}$.[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 [1] 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.
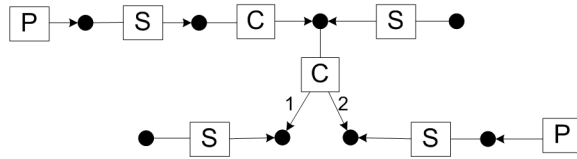
Fig. 4. Hypergraph defining a network with distributed processes

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

---

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

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.
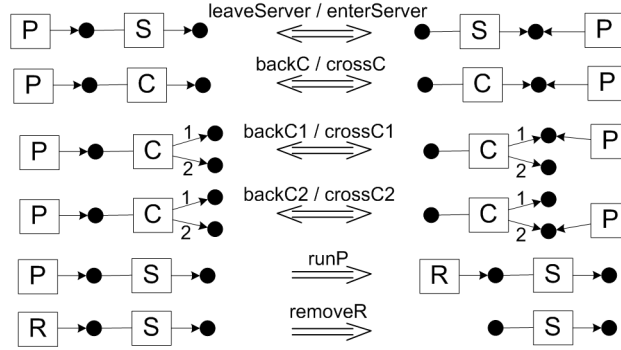

Fig. 5. Hypergraph transformation rules modelling the behaviour of mobile processes

Applying the $\mathcal{M}$-functor $\mathcal{F}_{HG}$ 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 $\mathcal{F}_{HG}$-reachable critical pair for the application of rules enterServer 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 enter a server location or to cross a connection channel in the network.
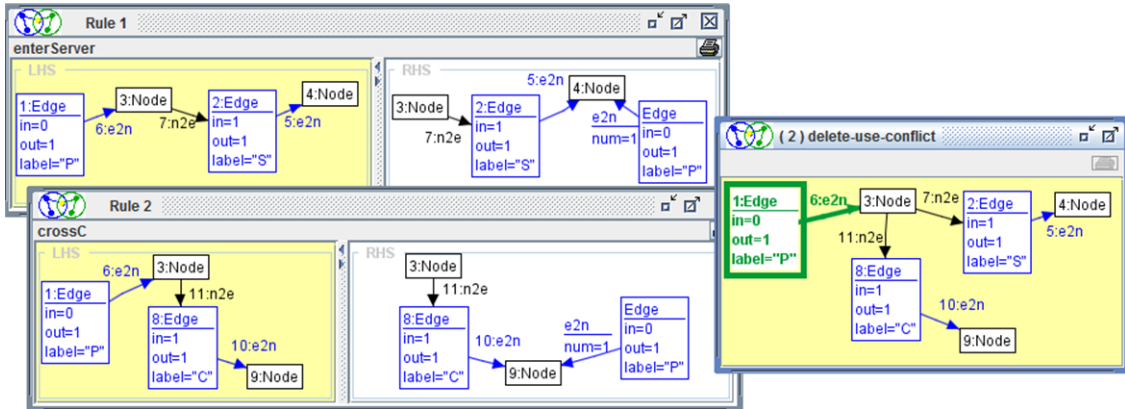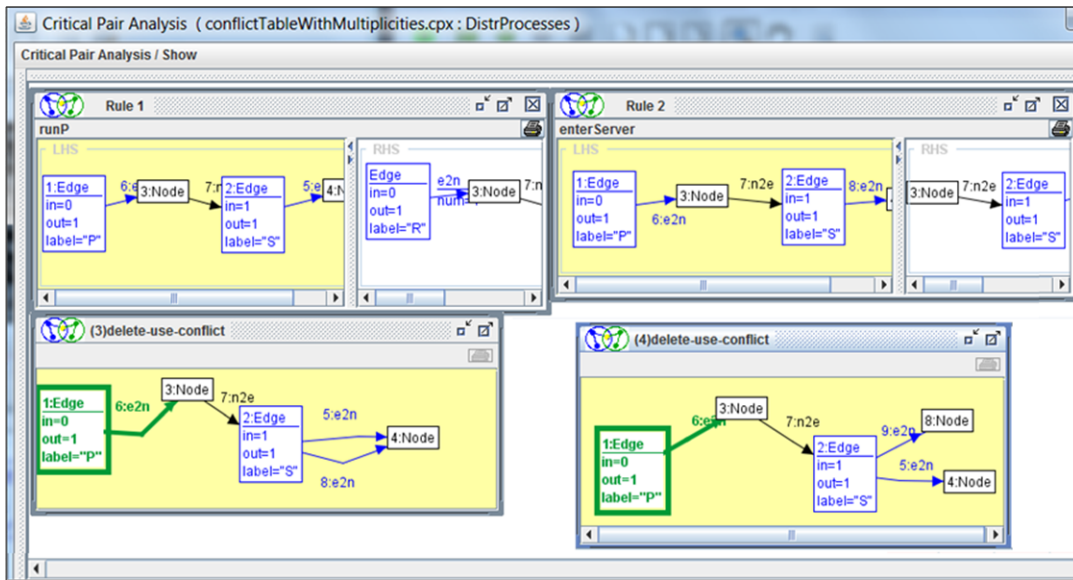

Fig. 6. A critical pair detected by AGG for the rule pair (enterServer, crossC)

This critical pair is locally 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

13

whole transformation system is locally confluent by applying Theorem 4, showing strict confluence of all $\mathcal{F}_{\mathrm{HG}}$-reachable critical pairs. 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.

Note that there are several non-$\mathcal{F}_{\mathrm{HG}}$-reachable critical pairs that do not have to be considered according to Theorem 4. Two examples for possible non-$\mathcal{F}_{\mathrm{HG}}$-reachable critical pairs constructed for the rule pair (runP, enterServer) are shown in Figure 7.

Fig. 7. Two non-$\mathcal{F}_{\mathrm{HG}}$-reachable critical pairs detected by AGG for rule pair (runP, enterServer)

The first delete-use-conflict (3) shows a non-$\mathcal{F}_{\mathrm{HG}}$-reachable overlapping graph, where a Server hyperedge has two outgoing edges connecting it to the same node. For this situation, an original hypergraph does not exist, since the attributes in=1 and out=1 mean that the original hypergraph has exactly one incoming and one outgoing edge. A similar overlapping graph is shown for the second delete-use-conflict (4), where also two outgoing edges exist, this time connecting the hyperedge with two different nodes.

6   Related Work and Conclusion

In our previous paper [7] we have developed a general framework to establish a formal relationship between different $\mathcal{M}$-adhesive transformation systems, showing

under which conditions transformations can be translated and created between different $\mathcal{M}$-adhesive transformation systems. This result is based on suitable properties of $\mathcal{M}$-functors between the corresponding $\mathcal{M}$-adhesive categories. In this paper, we construct an $\mathcal{M}$-functor from hypergraphs to typed attributed graphs. We show in our main technical result in Theorem 2 (with non-trivial proof in Appendix A) 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 [8]. This provides us with a general framework to analyse hypergraph transformation systems and allows us by Theorem 4 to use the critical pair analysis of the AGG-tool [12] 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 [1,2]. The thesis of D. Plump [11] 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 characterisation of $\mathcal{F}$-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 [5] can be handled in this framework in order to consider critical pairs and local confluence of $\mathcal{M}$-adhesive transformation systems with (nested) application conditions.

References

1. Baldan, P., Corradini, A., König, B.: Static analysis of distributed systems with mobility specified by graph grammars—a case study. In: Ehrig, H., Krämer, B., Ertas, A. (eds.) Proc. of Int. Conf. on Integrated Design & Process Technology. SDPS (2002)
2. Baldan, P., König, B.: Approximating the behaviour of graph transformation systems. In: Int. Conf. on Graph Transformation. pp. 14–29. Springer (2002), LNCS 2505
3. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in Theor. Comp. Science, Springer (2006)
4. Ehrig, H., Golas, U., Hermann, F.: Categorical Frameworks for Graph Transformation and HLR Systems based on the DPO Approach. EATCS Bulletin 102, 111–121 (2010)
5. Habel, A., Pennemann, K.H.: Nested constraints and application conditions for high-level structures. In: Kreowski, H.J., Montanari, U., Orejas, F., Rozenberg, G., Taentzer, G. (eds.) Formal Methods in Software and Systems Modeling. LNCS, vol. 3393, pp. 294–308. Springer (2005)
6. Lack, S., Sobociński, P.: Adhesive Categories. In: Proc. FOSSACS' 04. LNCS, vol. 2987, pp. 273–288. Springer (2004)
7. Maximova, M., Ehrig, H., Ermel, C.: Formal relationship between Petri net and graph transformation systems based on functors between $\mathcal{M}$-adhesive categories. In: Proc. of 4th Workshop on Petri Nets and Graph Transformation Systems. vol. 40. ECEASST (2011), http://journal.ub.tu-berlin.de/index.php/eceasst/issue/archive
8. Maximova, M., Ehrig, H., Ermel, C.: Transfer of local confluence and termination between Petri net and graph transformation systems based on $\mathcal{M}$-functors. In: Proc. of 5th Workshop on Petri Nets and Graph Transformation Systems. vol. 51, pp. 1–12. ECEASST (2012), http://journal.ub.tu-berlin.de/index.php/eceasst/issue/archive
9. Maximova, M., Ehrig, H., Ermel, C.: Analysis of Hypergraph Transformation Systems in AGG based on $\mathcal{M}$-Functors. ECEASST (2013), http://journal.ub.tu-berlin.de/index.php/eceasst/issue/archive, to appear
10. Minas, M.: Hypergraphs as a uniform diagram representation model. In: Proc. 6th Int. Workshop on Theory and Application of Graph Transformations (TAGT'98). LNCS, vol. 1764, pp. 281–295. Springer (2000)
11. Plump, D.: Evaluation of functional expressions by hypergraph rewriting. Ph.D. thesis, Universität Bremen, Fachbereich Mathematik und Informatik (1993)
12. TFS-Group, TU Berlin: AGG (2012), http://www.tfs.tu-berlin.de/agg

A Proofs

In this appendix we give the proofs for Lemma 1-4, where Lemma 1-3 are required to show that the $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathcal{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ (see Definition 4 in Section 3) translates and creates (direct) transformations (see Theorem 2 in Section 3) and Lemma 4 we are using to show that $\mathcal{F}_{\mathrm{HG}}$ creates local confluence based on $\mathcal{F}_{\mathrm{HG}}$-reachable critical pairs (see Theorem 4 in Section 4).

The following lemma shows that the functor $\mathcal{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathcal{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ preserves pushouts along $\mathcal{M}$-morphisms, which is a property to be satisfied by each $\mathcal{M}$-functor.

Lemma 1 ($\mathcal{F}_{\mathrm{HG}}$ Preserves Pushouts along $\mathcal{M}$-Morphisms).
*Consider hypergraphs G_i for $i \in \{0, 1, 2, 3\}$ with hypergraph morphisms $b = (b_V, b_E)$, $c = (c_V, c_E)$, $g = (g_V, g_E)$, $h = (h_V, h_E)$ and typed attributed graphs $\mathcal{F}_{\mathrm{HG}}(G_i)$ for $i \in \{0, 1, 2, 3\}$ with typed attributed graph morphisms $\mathcal{F}_{\mathrm{HG}}(b) = b' = (b'_{V_G}, b'_{V_D}, b'_{E_G}, b'_{E_{NA}}, b'_{E_{EA}})$, $\mathcal{F}_{\mathrm{HG}}(c) = c' = (c'_{V_G}, c'_{V_D}, c'_{E_G}, c'_{E_{NA}}, c'_{E_{EA}})$, $\mathcal{F}_{\mathrm{HG}}(g) = g' = (g'_{V_G}, g'_{V_D}, g'_{E_G}, g'_{E_{NA}}, g'_{E_{EA}})$, $\mathcal{F}_{\mathrm{HG}}(h) = h' = (h'_{V_G}, h'_{V_D}, h'_{E_G}, h'_{E_{NA}}, h'_{E_{EA}})$.
If (1) is a pushout in $\mathbf{HyperGraphs}$ with $b \in \mathcal{M}_1$, then we have that (2) is a pushout in $\mathbf{AGraphs_{HGTG}}$ with $\mathcal{F}_{\mathrm{HG}}(b) \in \mathcal{M}_2$.*

$$
\begin{array}{ccc}
G_0 \xrightarrow{\ b\ } G_1 & \qquad & \mathcal{F}_{\mathrm{HG}}(G_0) \xrightarrow{\mathcal{F}_{\mathrm{HG}}(b)=b'} \mathcal{F}_{\mathrm{HG}}(G_1) \\
c \downarrow \quad (1) \quad \downarrow g & & \mathcal{F}_{\mathrm{HG}}(c)=c' \downarrow \quad (2) \quad \downarrow \mathcal{F}_{\mathrm{HG}}(g)=g' \\
G_2 \xrightarrow{\ h\ } G_3 & & \mathcal{F}_{\mathrm{HG}}(G_2) \xrightarrow{\mathcal{F}_{\mathrm{HG}}(h)=h'} \mathcal{F}_{\mathrm{HG}}(G_3)
\end{array}
$$

Proof.
For the given morphism $b \in \mathcal{M}_1$ we have that $\mathcal{F}_{\mathrm{HG}}(b) = b' \in \mathcal{M}_2$, because $\mathcal{F}_{\mathrm{HG}}$ preserves injectivity of morphisms.

Given: (1) is a pushout in $\mathbf{HyperGraphs}$ with $b \in \mathcal{M}_1$, i.e., V- and E-components of (1) are pushouts in $\mathbf{Sets}$, because pushouts in $\mathbf{HyperGraphs}$ are constructed componentwise.

To show: V_G-, V_D-, E_G-, E_{NA}- and E_{EA}-components of (2) are pushouts in $\mathbf{Sets}$ with $\mathcal{F}_{\mathrm{HG}}(b) = b' \in \mathcal{M}_2$, because pushouts in $\mathbf{AGraphs_{HGTG}}$ are constructed componentwise as well.

1. V_G-component of (2) is a pushout in $\mathbf{Sets}$ (see diagram (3)) with $f'_{V_G} = f_V \uplus f_E$ for $f \in \{b, c, g, h\}$, because pushouts are compatible with coproducts.

2. V_D-component of (2) is obviously a pushout in $\mathbf{Sets}$ with $f'_{V_D} = id_{\mathbb{N}}$ for $f \in \{b, c, g, h\}$.

$$
\begin{array}{ccc}
V_G^{G_0} \xrightarrow{\ b_V \uplus b_E\ } V_G^{G_1} \\
c_V \uplus c_E \downarrow \quad (3) \quad \downarrow g_V \uplus g_E \\
V_G^{G_2} \xrightarrow{\ h_V \uplus h_E\ } V_G^{G_3}
\end{array}
$$

17

3. For the E_G-component we have to show that diagrams $(5a)$ and similarly $(5b)$ are pushouts. Diagrams $(5a)$ and $(5b)$ commute, because for each product component we have a pushout in **Sets** by assumption. So it remains to show that $(5a)$ and $(5b)$ are pushouts, because products of pushouts are not necessarily pushouts (the merging morphisms are in general not jointly surjective).

$$
\begin{array}{ccc}
E_{n2e}^{G_0} \xrightarrow{b_V \times b_E \times id_{\mathbb{N}}} E_{n2e}^{G_1} & \qquad & E_{e2n}^{G_0} \xrightarrow{b_E \times b_V \times id_{\mathbb{N}}} E_{e2n}^{G_1} \\
\downarrow{c_V \times c_E \times id_{\mathbb{N}}} \quad (5a) \quad \downarrow{g_V \times g_E \times id_{\mathbb{N}}} & & \downarrow \quad (5b) \quad \downarrow{g_E \times g_V \times id_{\mathbb{N}}} \\
E_{n2e}^{G_2} \xrightarrow[h_V \times h_E \times id_{\mathbb{N}}]{} E_{n2e}^{G_3} & & E_{e2n}^{G_2} \xrightarrow[h_E \times h_V \times id_{\mathbb{N}}]{} E_{e2n}^{G_3}
\end{array}
$$

with middle labels $c_E \times c_V \times id_{\mathbb{N}}$ for $(5a)$.

(a) For diagram $(5a)$ we have to show that the diagram $(5a')$ is a pushout in **Sets** with

$$
V_i \otimes E_i \otimes \mathbb{N} = \{(v, e, n) \in V_i \times E_i \times \mathbb{N} \mid s_i^n(e) = v\} \text{ for } i \in \{0, 1, 2, 3\}
$$
and $f_V \otimes f_E \otimes id_{\mathbb{N}}$ for $f \in \{b, c, g, h\}$.

$$
\begin{array}{ccc}
V_0 \otimes E_0 \otimes \mathbb{N} \xrightarrow{b_V \otimes b_E \otimes id_{\mathbb{N}}} V_1 \otimes E_1 \otimes \mathbb{N} \\
\downarrow{c_V \otimes c_E \otimes id_{\mathbb{N}}} \quad (5a') \quad \downarrow{g_V \otimes g_E \otimes id_{\mathbb{N}}} \\
V_2 \otimes E_2 \otimes \mathbb{N} \xrightarrow[h_V \otimes h_E \otimes id_{\mathbb{N}}]{} V_3 \otimes E_3 \otimes \mathbb{N}
\end{array}
$$

Since b, c, g, h are hypergraph morphisms, we have that all $f_V \otimes f_E \otimes id_{\mathbb{N}}$ morphisms for $f \in \{b, c, g, h\}$ are well-defined. Furthermore, we have that the components of $(5a')$ are pushouts and pullbacks in **Sets**, because $b = (b_V, b_E) \in \mathcal{M}_1$ by assumption. Hence, also $(5a')$ is a pullback, because pullbacks are compatible with products and it remains to show that $(h_V \otimes h_E \otimes id_{\mathbb{N}}, g_V \otimes g_E \otimes id_{\mathbb{N}})$ are jointly surjective.

Consider $(v_3, e_3, n) \in V_3 \otimes E_3 \otimes \mathbb{N}$. The E-component of $(5a')$ is a pushout in **Sets** s.t. we have $e_1 \in E_1$ with $g_E(e_1) = e_3$ (or we have $e_2 \in E_2$ with $h_E(e_2) = e_3$). Without loss of generality we consider the first case. Let $v_1 = s_1^n(e_1)$ for $n \in \mathbb{N}$, then $(g_V \otimes g_E \otimes id_{\mathbb{N}})(v_1, e_1, n) = (v_3, e_3, n)$, because $g = (g_V, g_E)$ is a hypergraph morphism, i.e., the compatibility with the corresponding source function holds s.t. we have $s_3^n(e_3) = v_3$. Hence, $(5a')$ and $(5a)$ are pushouts.

(b) For diagram $(5b)$ the proof is similar to the Case 3a switching the components of the product.

4. For the E_{NA}-component we have to show that diagrams $(6a)$ and similarly $(6b)$ are pushouts for $X = \{in\}$ and $Y = \{out\}$. Diagrams $(6a)$ and $(6b)$ commute, because for each product component we have a pushout in **Sets** by assumption. It can be shown similar to the Case 3 that $(6a)$ and $(6b)$ are pushouts.

$$E_{in}^{G_0} \xrightarrow{\ b_E \times id_\mathbb{N} \times id_X\ } E_{in}^{G_1}$$

$$\left\downarrow{\scriptstyle c_E \times id_\mathbb{N} \times id_X}\right. \qquad (6a) \qquad \left\downarrow{\scriptstyle g_E \times id_\mathbb{N} \times id_X}\right.$$

$$E_{in}^{G_2} \xrightarrow{\ h_E \times id_\mathbb{N} \times id_X\ } E_{in}^{G_3}$$

$$E_{out}^{G_0} \xrightarrow{\ b_E \times id_\mathbb{N} \times id_Y\ } E_{out}^{G_1}$$

$$\left\downarrow{\scriptstyle c_E \times id_\mathbb{N} \times id_Y}\right. \qquad (6b) \qquad \left\downarrow{\scriptstyle g_E \times id_\mathbb{N} \times id_Y}\right.$$

$$E_{out}^{G_2} \xrightarrow{\ h_E \times id_\mathbb{N} \times id_Y\ } E_{out}^{G_3}$$

5. The proof for the E_{EA}-component is similar to that of the E_G-component in the Case 3 by switching the components of the products.

According to the following lemma, each typed attributed graph morphism $f' : \mathcal{F}_{\mathrm{HG}}(G_1) \to \mathcal{F}_{\mathrm{HG}}(G_2)$ is uniquely determined by its V_G-component $f'_{V_G} : V_G^{G_1} \to V_G^{G_2}$. We are using this property to show the uniqueness of morphism creation.

Lemma 2 (Uniquely Determined $\mathcal{F}_{\mathrm{HG}}$-Images).
Consider a typed attributed graph morphism $f' : \mathcal{F}_{\mathrm{HG}}(G_1) \to \mathcal{F}_{\mathrm{HG}}(G_2)$ with $f'_{V_D} = id_\mathbb{N}$.
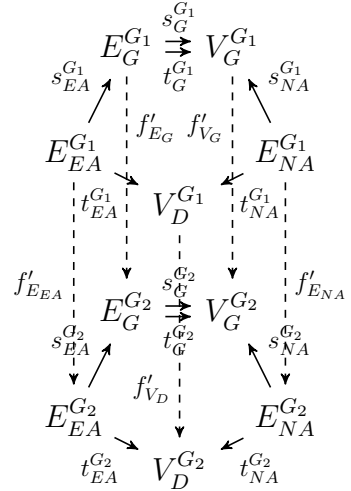*Then we have that f' is uniquely determined by the component $f'_{V_G} : V_G^{G_1} \to V_G^{G_2}$ with $V_G^{G_i} = V_{G_i} \uplus E_{G_i}$ for $i \in \{1, 2\}$.*

Proof.
Given a typed attributed graph morphism $f' : \mathcal{F}_{\mathrm{HG}}(G_1) \to \mathcal{F}_{\mathrm{HG}}(G_2)$ with $f' = (f'_{V_G}, f'_{V_D} = id_\mathbb{N}, f'_{E_G}, f'_{E_{NA}}, f'_{E_{EA}})$ (see diagram below) and $f_V(v) = f'_{V_G}(v)$ for $v \in V_{G_1}$, $f_E(e) = f'_{V_G}(e)$ for $e \in E_{G_1}$.

To show:

1. $f'_{E_G}(v, e, n) = (f_V(v), f_E(e), n)$ for $(v, e, n) \in E_{n2e}^{G_1}$,
2. $f'_{E_G}(e, v, n) = (f_E(e), f_V(v), n)$ for $(e, v, n) \in E_{e2n}^{G_1}$,
3. $f'_{E_{NA}}(e, n, x) = (f_E(e), n, x)$ for $(e, n, x) \in E_{in}^{G_1} \uplus E_{out}^{G_1}$,
4. $f'_{E_{EA}}(n, v, e) = (n, f_V(v), f_E(e))$ for $(n, v, e) \in E_s^{G_1}$,
5. $f'_{E_{EA}}(n, e, v) = (n, f_E(e), f_V(v))$ for $(n, e, v) \in E_t^{G_1}$.

$$
\begin{array}{ccc}
 & E_G^{G_1} \overset{s_G^{G_1}}{\underset{t_G^{G_1}}{\rightrightarrows}} V_G^{G_1} & \\
{\scriptstyle s_{EA}^{G_1}}\nearrow \quad f'_{E_G}\ f'_{V_G} \quad \nwarrow{\scriptstyle s_{NA}^{G_1}} & & \\
E_{EA}^{G_1} & & E_{NA}^{G_1} \\
{\scriptstyle t_{EA}^{G_1}}\downarrow \quad V_D^{G_1} \quad \downarrow{\scriptstyle t_{NA}^{G_1}} & & \\
f'_{E_{EA}}\downarrow \quad E_G^{G_2} \overset{s_G^{G_2}}{\underset{t_G^{G_2}}{\rightrightarrows}} V_G^{G_2} \quad \downarrow f'_{E_{NA}} & & \\
{\scriptstyle s_{EA}^{G_2}}\nearrow \quad f'_{V_D} \quad \searrow{\scriptstyle s_{NA}^{G_2}} & & \\
E_{EA}^{G_2} & & E_{NA}^{G_2} \\
{\scriptstyle t_{EA}^{G_2}}\searrow \quad V_D^{G_2} \quad \swarrow{\scriptstyle t_{NA}^{G_2}} & &
\end{array}
$$

Case 1:
Let $f'_{E_G}(v, e, n) = (v', e', n')$ with $s_G^{G_2}(v', e', n') = v'$ and $t_G^{G_2}(v', e', n') = e'$. Then

19

we have:

$$v' = s_G^{G_2}(v', e', n') = s_G^{G_2}(f'_{E_G}(v, e, n)) = f'_{V_G}(s_G^{G_1}(v, e, n)) = f'_{V_G}(v) = f_V(v)$$
$$\Rightarrow v' = f_V(v),$$
$$e' = t_G^{G_2}(v', e', n') = t_G^{G_2}(f'_{E_G}(v, e, n)) = f'_{V_G}(t_G^{G_1}(v, e, n)) = f'_{V_G}(e) = f_E(e)$$
$$\Rightarrow e' = f_E(e),$$
$$n = id_{\mathbb{N}}(t_{EA}^{G_1}(n, v, e)) = t_{EA}^{G_2}(f'_{E_{EA}}(n, v, e)) \overset{(*)}{=} t_{EA}^{G_2}(n', v', e') = n' \Rightarrow n' = n$$

$$\Rightarrow f'_{E_G}(v, e, n) = (v', e', n') = (f_V(v), f_E(e), n)$$

$$(*): s_{EA}^{G_2}(f'_{E_{EA}}(n, v, e)) = f'_{E_G}(s_{EA}^{G_1}(n, v, e)) = f'_{E_G}(v, e, n)$$
$$= (v', e', n') = s_{EA}^{G_2}(n', v', e')$$
$$\Rightarrow f'_{E_{EA}}(n, v, e) = (n', v', e') \text{ since } s_{EA}^{G_2} \text{ is a bijection and therefore also an}$$
injection

Cases 2-5: Similar to the Case 1.

The following lemma with its detailed proof states the fact that the $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}} : (\mathbf{HyperGraphs}, \mathcal{M}_1) \to (\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ given in Definition 4 in Section 3 preserves initial pushouts, i.e., $(\mathbf{HyperGraphs}, \mathcal{M}_1)$ has initial pushouts, which become also initial pushouts in $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ after application of $\mathcal{F}_{\mathrm{HG}}$.

Lemma 3 (Preservation of Initial Pushouts).
Let (1) *be an initial pushout over* $f : L \to G$ *in* $(\mathbf{HyperGraphs}, \mathcal{M}_1)$. *Then* (2) *is an initial pushout over* $\mathcal{F}_{\mathrm{HG}}(f) : \mathcal{F}_{\mathrm{HG}}(L) \to \mathcal{F}_{\mathrm{HG}}(G)$ *in* $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$.

$$
\begin{array}{ccc}
B \xrightarrow{\ b\ } L & & \mathcal{F}_{\mathrm{HG}}(B) \xrightarrow{\mathcal{F}_{\mathrm{HG}}(b)} \mathcal{F}_{\mathrm{HG}}(L) \\
\downarrow \quad (1) \quad \downarrow f & \Longrightarrow & \downarrow \quad (2) \quad \downarrow \mathcal{F}_{\mathrm{HG}}(f) \\
C \longrightarrow G & & \mathcal{F}_{\mathrm{HG}}(C) \longrightarrow \mathcal{F}_{\mathrm{HG}}(G)
\end{array}
$$

Proof.
Similar to initial pushouts in $(\mathbf{Graphs}, \mathcal{M})$ (see [3]) we also have initial pushouts in $(\mathbf{HyperGraphs}, \mathcal{M}_1)$ using a boundary construction based on dangling and identification points.
We assume that (1) is an initial pushout in $(\mathbf{HyperGraphs}, \mathcal{M}_1)$ with boundary

$B = (V_B, E_B, s_B = s_L, t_B = t_L) \subseteq L$, where

> $V_B = DP_V \cup IP_V \cup IP_{VE}$ with dangling points
> > $DP_V = \{v \in V_L \mid \exists e \in E_G \backslash f_E(E_L).(f_V(v) \Subset s_G(e)) \ \vee \ (f_V(v) \Subset t_G(e))\}$,
> > where $x \Subset w \Leftrightarrow \exists w_1, w_2. \ w = w_1 \cdot x \cdot w_2$ and identification points
> > $IP_V = \{v \in V_L \mid \exists v' \neq v.v' \in V_L \ \wedge \ f_V(v) = f_V(v')\}$,
> > $IP_{VE} = \{\overline{v} \in V_L \mid \exists e \in IP_E. \ \overline{v} \Subset s_L(e) \ \vee \ \overline{v} \Subset t_L(e)\}$,
> $E_B = IP_E = \{e \in E_L \mid \exists e' \neq e.e' \in E_L \ \wedge \ f_E(e) = f_E(e')\}$

Note that IP_{VE} is needed in order to make sure that B is a well-defined hypergraph to avoid an intersection construction for B.

Since $\mathcal{F}_{HG}$ preserves pushouts along $\mathcal{M}$-morphisms (see Lemma 1 in this appendix) we have that (2) is a pushout in $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ with $\mathcal{F}_{HG}(b) \in \mathcal{M}_2$.

Now we construct similar to [3] the initial pushout (3) over $\mathcal{F}_{HG}(f) : \mathcal{F}_{HG}(L) \to \mathcal{F}_{HG}(G)$ in $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ with inclusion $b' : B' \to \mathcal{F}_{HG}(L)$ and boundary $B' = ((B'_0, NAT), type)$, where the components of $B'_0 = (V_G^{B'_0}, V_D^{B'_0} = \mathbb{N}, E_G^{B'_0}, E_{NA}^{B'_0},$
$E_{EA}^{B'_0}, (s_j^{B'_0}, t_j^{B'_0})_{j \in \{G, NA, EA\}})$ are defined as follows:

$E_{NA}^{B'_0} = IP_{E_{NA}} = \{a \in E_{NA}^{L'} = E_{in}^{L'} \uplus E_{out}^{L'} \mid \exists a' \neq a. \ a' \in E_{NA}^{L'} = E_{in}^{L'} \uplus E_{out}^{L'} \ \wedge$
$f'_{E_{NA}}(a) = f'_{E_{NA}}(a')\}$, where $f'_{E_{NA}}$ is defined as given in Definition 4,
$E_{EA}^{B'_0} = IP_{E_{EA}} = \{a \in E_{EA}^{L'} = E_s^{L'} \uplus E_t^{L'} \mid \exists a' \neq a. \ a' \in E_{EA}^{L'} = E_s^{L'} \uplus E_t^{L'} \ \wedge$
$f'_{E_{EA}}(a) = f'_{E_{EA}}(a')\}$, where $f'_{E_{EA}}$ is defined as given in Definition 4,
$V_G^{B'_0} = DP_{V_G} \cup IP_{V_G} \cup s_G^{L'}(IP_{E_G}) \cup t_G^{L'}(IP_{E_G}) \cup s_{NA}^{L'}(IP_{E_{NA}})$ with

$DP_{V_G} = \{a \in V_G^{L'} = V_L \uplus E_L \mid [\exists a' \in E_{NA}^{G'} \backslash f'_{E_{NA}}(E_{NA}^{L'}) = (E_{in}^{G'} \uplus E_{out}^{G'}) \backslash$
$\qquad f'_{E_{NA}}(E_{in}^{L'} \uplus E_{out}^{L'}). \ f'_{V_G}(a) = s_{NA}^{G'}(a')] \vee [\exists a' \in E_G^{G'} \backslash f'_{E_G}(E_G^{L'}) =$
$\qquad (E_{n2e}^{G'} \uplus E_{e2n}^{G'}) \backslash f'_{E_G}(E_{n2e}^{L'} \uplus E_{e2n}^{L'}). \ f'_{V_G}(a) = s_G^{G'}(a') \vee f'_{V_G}(a) = t_G^{G'}(a')]\}$,
$\qquad$ where $f'_{E_{NA}}, f'_{V_G}$ and f'_{E_G} are defined as given in Definition 4,

$$IP_{V_G} = \{a \in V_G^{L'} = V_L \uplus E_L \mid \exists a' \neq a. \; a' \in V_G^{L'} = V_L \uplus E_L \; \wedge \; f'_{V_G}(a) = f'_{V_G}(a')\},$$
$$\text{where } f'_{V_G} \text{ is defined as given in Definition 4,}$$

$$E_G^{B'_0} = DP_{E_G} \cup IP_{E_G} \cup s_{EA}^{L'}(IP_{E_{EA}}) \text{ with}$$

$$DP_{E_G} = \{a \in E_G^{L'} = E_{n2e}^{L'} \uplus E_{e2n}^{L'} \mid \exists a' \in E_{EA}^{G'} \backslash f'_{E_{EA}}(E_{EA}^{L'}) = (E_s^{G'} \uplus E_t^{G'}) \backslash$$
$$f'_{E_{EA}}(E_s^{L'} \uplus E_t^{L'}). \; f'_{E_G}(a) = s_{EA}^{G'}(a')\},$$
$$\text{where } f'_{E_{EA}} \text{ and } f'_{E_G} \text{ are defined as given in Definition 4,}$$
$$IP_{E_G} = \{a \in E_G^{L'} = E_{n2e}^{L'} \uplus E_{e2n}^{L'} \mid \exists a' \neq a. \; a' \in E_G^{L'} = E_{n2e}^{L'} \uplus E_{e2n}^{L'} \wedge \; f'_{E_G}(a) =$$
$$f'_{E_G}(a')\}, \text{ where } f'_{E_G} \text{ is defined as given in Definition 4,}$$

and source/target functions in B'_0 are restrictions of the corresponding functions in $\mathcal{F}_{\mathrm{HG}}(L) = L'$.

Initiality of (3) implies unique morphisms $i : B' \to \mathcal{F}_{\mathrm{HG}}(B)$ and $j : C' \to \mathcal{F}_{\mathrm{HG}}(C)$ such that (4) is a pushout in $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ and (5), (6) commute with $i \in \mathcal{M}_2$.

Consider the typed attributed graph $\mathcal{F}_{\mathrm{HG}}(B) = B^* = ((B_0^*, NAT), type)$ with $B_0^* = (V_G^{B_0^*}, V_D^{B_0^*} = \mathbb{N}, E_G^{B_0^*}, E_{NA}^{B_0^*}, E_{EA}^{B_0^*}, (s_j^{B_0^*}, t_j^{B_0^*})_{j \in \{G, NA, EA\}})$ defined according to Definition 4 in Section 3.

It remains to show: $i : B' \to \mathcal{F}_{\mathrm{HG}}(B)$ is surjective, i.e., $\mathcal{F}_{\mathrm{HG}}(B) = B^* \subseteq B'$, i.e., $V_G^{B_0^*} \subseteq V_G^{B'_0} \; \wedge \; E_G^{B_0^*} \subseteq E_G^{B'_0} \; \wedge \; E_{NA}^{B_0^*} \subseteq E_{NA}^{B'_0} \; \wedge \; E_{EA}^{B_0^*} \subseteq E_{EA}^{B'_0}$.

In the following we prove only the first conjunct. The proofs for the other conjuncts can be done in the similar way.

To show: $V_G^{B_0^*} \subseteq V_G^{B'_0}$.
From Definition 4 in Section 3 and constructions of B, B' above we have:

$$V_G^{B_0^*} = V_B \uplus E_B = IP_V \cup IP_{VE} \cup DP_V \cup IP_E \text{ and}$$
$$V_G^{B'_0} = DP_{V_G} \cup IP_{V_G} \cup s_G^{L'}(IP_{E_G}) \cup t_G^{L'}(IP_{E_G}) \cup s_{NA}^{L'}(IP_{E_{NA}})$$

1. Let $v \in DP_V$. Then we consider two cases:
   (a) $v \in V_L$ and $\exists e \in E_G \backslash f_E(E_L). \; f_V(v) \in s_G(e)$, i.e., $\exists n \in \mathbb{N}. \; f_V(v) = s_G^n(e)$.
   It remains to show: $v \in DP_{V_G} \subseteq V_G^{B'_0}$.
   From $v \in V_L$ we get that $v \in V_G^{L'}$. Furthermore, let $a' = (f_V(v), e, n) \in E_{n2e}^{G'}$, because $f_V(v) = s_G^n(e)$. Then we have:
   $f'_{V_G}(v) = f_V(v) = s_G^{G'}(a')$ and $e \in E_G \backslash f_E(E_L) \Rightarrow a' = (f_V(v), e, n) \notin f'_{E_G}(E_G^{L'})$
   $\overset{Constr. \; B'}{\Rightarrow} v \in DP_{V_G} \subseteq V_G^{B'_0}.$

22

(b) $v \in V_L$ and $\exists e \in E_G \backslash f_E(E_L)$. $f_V(v) \in t_G(e)$, i.e., $\exists n \in \mathbb{N}.\ f_V(v) = t_G^n(e)$:
Similar to the Case 1a above.

2. Let $v \in IP_{VE}$. Then we consider two cases:
   (a) $v \in V_L$ and $\exists e \in IP_E.\ v \in s_L(e)$, i.e., $\exists n \in \mathbb{N}.\ v = s_L^n(e)$.

   It remains to show: $v \in s_G^{L'}(IP_{E_G}) \subseteq V_G^{B_0'}$.
   From $e \in IP_E$ we get that $\exists e' \neq e.\ e' \in E_L\ \wedge\ f_E(e) = f_E(e')$.
   Let $a = (v, e, n) \in E_{n2e}^{L'}$ and we construct $a' \neq a$ with $a' = (v', e', n) \in E_{n2e}^{L'}$
   s.t. $v' = s_L^n(e')$. Then we have:

   $$f_V(v) = f_V(s_L^n(e)) = [f_V^*(s_L(e))]_n = [s_G(f_E(e))]_n$$
   $$= [s_G(f_E(e'))]_n = [f_V^*(s_L(e'))]_n = f_V(s_L^n(e'))$$
   $$= f_V(v'), \text{ where } [\ ]_n \text{ denotes } n\text{-th component of } [\ ]$$

   This implies:
   $f_{E_G}'(a) = f_{E_G}'(v, e, n) = (f_V(v), f_E(e), n) = (f_V(v'), f_E(e'), n)$
   $= f_{E_G}'(v', e', n) = f_{E_G}'(a')$

   $\overset{Constr.\ B'}{\Rightarrow} a \in IP_{E_G}$
   $\Rightarrow v \in s_G^{L'}(IP_{E_G})$ with $v = s_G^{L'}(a)$
   $\Rightarrow v \in s_G^{L'}(IP_{E_G}) \subseteq V_G^{B_0'}$.

$$
\begin{array}{ccc}
E_L & \underset{t_L}{\overset{s_L}{\rightrightarrows}} & V_L^* \\
f_E \downarrow & = & \downarrow f_V^* \\
E_G & \underset{t_G}{\overset{s_G}{\rightrightarrows}} & V_G^*
\end{array}
$$

   (b) $v \in V_L$ and $\exists e \in IP_E.\ v \in t_L(e)$, i.e., $\exists n \in \mathbb{N}.\ v = t_L^n(e)$: Similar to the
   Case 2a above.

3. Let $v \in IP_V$. Then we have: $v \in V_L$ and $\exists v' \neq v.\ v' \in V_L\ \wedge\ f_V(v) = f_V(v')$.
   It remains to show: $v \in IP_{V_G} \subseteq V_G^{B_0'}$.
   From $v, v' \in V_L$ we get that $v, v' \in V_G^{L'}$. Furthermore, we have:

   $f_V(v) = f_V(v') \Rightarrow f_{V_G}'(v) = f_V(v) = f_V(v') = f_{V_G}'(v') \overset{Constr.\ B'}{\Rightarrow} v \in IP_{V_G} \subseteq V_G^{B_0'}$.

4. Let $e \in IP_E$. Then we have: $e \in E_L$ and $\exists e' \neq e.\ e' \in E_L\ \wedge\ f_E(e) = f_E(e')$.
   We can show that $e \in IP_{V_G} \subseteq V_G^{B_0'}$ similar to the Case 3 above.

This concludes the part of surjectivity of $i : B' \to \mathcal{F}_{\mathrm{HG}}(B)$. Now i is injective and surjective, so we get that i is an isomorphism. Since (4) is a pushout, also $j : C' \to \mathcal{F}_{\mathrm{HG}}(C)$ is an isomorphism and hence (2) is isomorphic to (3). So we get that also (2) is an initial pushout over $\mathcal{F}_{\mathrm{HG}}(f) : \mathcal{F}_{\mathrm{HG}}(L) \to \mathcal{F}_{\mathrm{HG}}(G)$.

According to the following lemma, the $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}} : (\textbf{HyperGraphs}, \mathcal{M}_1) \to (\textbf{AGraphs}_{\textbf{HGTG}}, \mathcal{M}_2)$ given in Definition 4 in Section 3 is compatible with pair factorisation, which is needed to prove our main conceptual result (Theorem 4 in Section 4).

Lemma 4 ($\mathcal{F}_{\mathrm{HG}}$ is Compatible with Pair Factorisation).
The $\mathcal{M}$-functor $\mathcal{F}_{\mathrm{HG}} : (\textbf{HyperGraphs}, \mathcal{M}_1) \to (\textbf{AGraphs}_{\textbf{HGTG}}, \mathcal{M}_2)$ is compatible with pair factorisation, i.e., (\textbf{HyperGraphs}, \mathcal{M}_1) and (\textbf{AGraphs}_{\textbf{HGTG}},

$\mathcal{M}_2$) *have pair factorisation and $\mathcal{F}_{\mathrm{HG}}$ preserves pair factorisation. In more detail, $\mathcal{F}_{\mathrm{HG}}$ preserves pair factorisation if for each pair factorisation $(f_1 = m \circ e_1, f_2 = m \circ e_2)$ in $(\mathbf{HyperGraphs}, \mathcal{M}_1)$ also $(\mathcal{F}_{\mathrm{HG}}(f_1) = \mathcal{F}_{\mathrm{HG}}(m) \circ \mathcal{F}_{\mathrm{HG}}(e_1), \mathcal{F}_{\mathrm{HG}}(f_2) = \mathcal{F}_{\mathrm{HG}}(m) \circ \mathcal{F}_{\mathrm{HG}}(e_2))$ is a pair factorisation in $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$ (see the diagram below).*

Proof.
Pair factorisation of $(f_1 : L_1 \to G, f_2 : L_2 \to G)$ based on $\mathcal{E}_1 - \mathcal{M}_1$-factorisation (and coproducts) is given by $(f_1 = m \circ e_1, f_2 = m \circ e_2)$, where $f : L_1 + L_2 \to G$ is the induced morphism of $f_i : L_i \to G$ for $i \in \{1, 2\}$, $f = m \circ e$ is an $\mathcal{E}_1 - \mathcal{M}_1$-factorisation, and $e_1 = e \circ i_1$, $e_2 = e \circ i_2$ are defined via the coproduct morphisms $i_1 : L_1 \to L_1 + L_2$ and $i_2 : L_2 \to L_1 + L_2$ in $(\mathbf{HyperGraphs}, \mathcal{M}_1)$. Similar we obtain a pair factorisation in $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$, and $\mathcal{E}_1$, $\mathcal{E}_2$ are classes of surjective morphisms leading to the classical $\mathcal{E}_i - \mathcal{M}_i$-factorisation for $i \in \{1, 2\}$. In order to show that $\mathcal{F}_{\mathrm{HG}}$ is compatible with pair factorisation it remains to show that $\mathcal{F}_{\mathrm{HG}}$ preserves pair factorisation, i.e., for each pair factorisation $(f_1 = m \circ e_1, f_2 = m \circ e_2)$ in $(\mathbf{HyperGraphs}, \mathcal{M}_1)$ also $(\mathcal{F}_{\mathrm{HG}}(f_1) = \mathcal{F}_{\mathrm{HG}}(m) \circ \mathcal{F}_{\mathrm{HG}}(e_1)$, $\mathcal{F}_{\mathrm{HG}}(f_2) = \mathcal{F}_{\mathrm{HG}}(m) \circ \mathcal{F}_{\mathrm{HG}}(e_2))$ is a pair factorisation in $(\mathbf{AGraphs_{HGTG}}, \mathcal{M}_2)$. But this can be concluded, if $\mathcal{F}_{\mathrm{HG}}$ preserves coproducts and satisfies $\mathcal{F}_{\mathrm{HG}}(\mathcal{E}_1) \subseteq \mathcal{E}_2$ as well as $\mathcal{F}_{\mathrm{HG}}(\mathcal{M}_1) \subseteq \mathcal{M}_2$. The first property can be shown similar to Lemma 1 in this appendix. $\mathcal{F}_{\mathrm{HG}}(\mathcal{E}_1) \subseteq \mathcal{E}_2$ means that $\mathcal{F}_{\mathrm{HG}}$ preserves surjectivity of morphisms, which is shown below, and $\mathcal{F}_{\mathrm{HG}}(\mathcal{M}_1) \subseteq \mathcal{M}_2$ is valid, because $\mathcal{F}_{\mathrm{HG}}$ is an $\mathcal{M}$-functor.
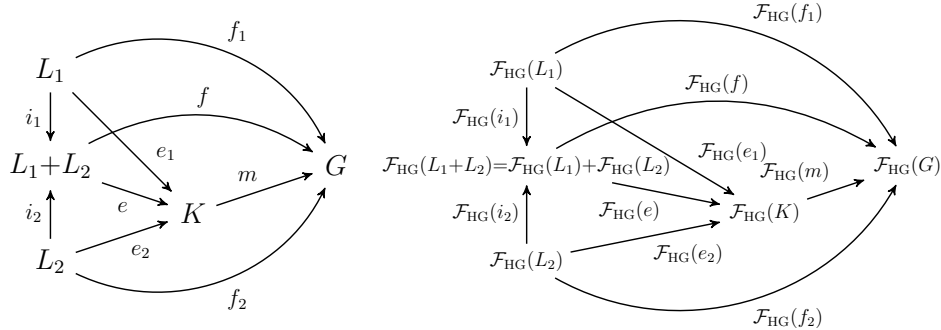
It remains to show that $\mathcal{F}_{\mathrm{HG}}$ preserves surjectivity of morphisms.
Let $f = (f_V, f_E)$ be a surjective hypergraph morphism, i.e., $\forall v' \in V_{G_2}.$ $\exists v \in V_{G_1}.$ $v' = f_V(v)$ and $\forall e' \in E_{G_2}.$ $\exists e \in E_{G_1}.$ $e' = f_E(e)$.
We have to show that the corresponding typed attributed graph morphism $\mathcal{F}_{\mathrm{HG}}(f) = f' = (f'_{V_G}, f'_{V_D}, f'_{E_G}, f'_{E_{NA}}, f'_{E_{EA}})$ is also surjective.
1. To show: f'_{V_G} is surjective, i.e., $\forall x' \in V_G^{G_2}.$ $\exists x \in V_G^{G_1}.$ $(x' = f_V(x)) \lor (x' = f_E(x))$.
   $f'_{V_G} : V_G^{G_1} \to V_G^{G_2}$ with $f'_{V_G} = f_V \uplus f_E$ for $V_G^{G_i} = V_{G_i} \uplus E_{G_i}$ is surjective, because the components f_V and f_E are surjective.

24

2. To show: f'_{V_D} is surjective, i.e., $\forall n' \in \mathbb{N}. \exists n \in \mathbb{N}. n' = id_{\mathbb{N}}(n)$. This holds obviously.

3. To show: f'_{E_G} is surjective.
$f'_{E_G} : E_G^{G_1} \to E_G^{G_2}$ for $E_G^{G_i} = E_{n2e}^{G_i} \uplus E_{e2n}^{G_i}$,
where $E_{n2e}^{G_i} = \{(v, e, n) \in (V_{G_i} \times E_{G_i}) \times \mathbb{N} \mid s_{G_i}^n(e) = v\}$
and $E_{e2n}^{G_i} = \{(e, v, n) \in (E_{G_i} \times V_{G_i}) \times \mathbb{N} \mid t_{G_i}^n(e) = v\}$.

$$
\begin{array}{ccc}
E_{G_1} & \xrightarrow[t_{G_1}]{s_{G_1}} & V_{G_1}^* \\
f_E \downarrow & = & \downarrow f_V^* \\
E_{G_2} & \xrightarrow[t_{G_2}]{s_{G_2}} & V_{G_2}^*
\end{array}
$$

(a) Let $(v', e', n') \in E_{n2e}^{G_2}$ with $s_{G_2}^{n'}(e') = v'$.
It remains to show: $\exists(v, e, n) \in E_{n2e}^{G_1}. f'_{E_G}(v, e, n) = (v', e', n')$.
Because of surjectivity of f_E holds: $\exists e \in E_{G_1}. e' = f_E(e)$ and let $v = s_{G_1}^{n'}(e)$.
Then we get, that $(v, e, n') \in E_{n2e}^{G_1}$, because $s_{G_1}^{n'}(e) = v$. Furthermore, we have:

$$
\begin{aligned}
v' = s_{G_2}^{n'}(e') & \stackrel{f_E \ surj.}{=} s_{G_2}^{n'}(f_E(e)) = [s_{G_2}(f_E(e))]_{n'} = [f_V^*(s_{G_1}(e))]_{n'} \\
& = f_V(s_{G_1}^{n'}(e)) = f_V(v), \text{ where } [\ ]_n \text{ denotes } n\text{-th component of } [\ ]
\end{aligned}
$$

This implies: $f'_{E_G}(v, e, n') = (f_V(v), f_E(e), n') = (v', e', n')$.

(b) Let $(e', v', n') \in E_{e2n}^{G_2}$ with $t_{G_2}^{n'}(e') = v'$: Similar to the case above.
We can show that $f'_{E_{NA}}$ and $f'_{E_{EA}}$ are surjective similar to the Case 3.