Distributed Graphs and Graph Transformation

Gabriele Taentzer
Technical University of Berlin, Germany

Abstract. The new approach of distributed graphs and graph transformation as developed in this article allows to use structured graph transformation on two abstraction levels, the network and the local level. The network level contains the description of the topological structures of a system. The local level covers the description of states and their transitions in local systems. Local state transitions may depend on others using suitable synchronization mechanisms. The main distribution concepts of categorical graph grammars presented by Schneider are combined with the algebraic approach to distributed graph transformation introduced by Ehrig et.al.

Modeling of distributed systems by this new kind of distributed graph transformation offers a clear and elegant description of dynamic networks, distributed actions as well as communication and synchronization using a graphical notation. Moreover, distributed graph transformation offers the possibility to describe splitting and joining of local graphs as well as parallel transformations in local systems.

The formalization of distributed graph transformation is done by means of category theory. A distributed graph is formalized by a diagram in the category GRAPH of graphs and total graph morphisms. A distributed transformation step is characterized by a double-pushout in the category DISTR(GRAPH) of distributed graphs and distributed graph morphisms. A pushout over distributed graph morphisms cannot always be constructed componentwise in each local part. But especially the componentwise construction of a distributed transformation reflects distributed computations best. Thus, we present the necessary conditions for componentwise construction of pushouts in DISTR(GRAPH) and show that they are also sufficient. These conditions are summarized in the distributed gluing condition. Moreover, these conditions are needed for componentwise construction of pushout complements which are used to characterize the first step of a distributed graph transformation.

1. Introduction

Graphical representations are an obvious means to describe different aspects of systems. Modeling distributed and concurrent systems graphs are often used to describe the *topological structure* of the system. The graphical structure shows then which parts are involved and what are the ways of communication. Graph transformations can be used conveniently to model dynamic changes of the system structure. For example, the distribution of some local parts is rearranged or communication channels are created or deleted. Local states are typically coded in some specification or programming text or not considered. This idea is followed, for example, in (DM87), by Δ-grammars in (KLG91), in (Sch92) and by actor graph grammars in (JR91; Kor94).

Graphs can be used also to model complex *object relations* inside of local parts of a system as they arise, for example, in software process modeling (like project flow graphs in (KH95) or development graphs in (PW94)). Graph transformations are useful then on these lower levels to specify changes of object relations.

Existing graph transformation models for distributed systems mostly concentrate on the topological aspects of a distributed system, e.g. (DM87; JR91; KLG91). In contrast, the algebraic approach to distributed graph transformation (EBHL88; EL93) describes mostly local actions by graph transformation related to others by interface graphs. The operations "split" and "join" switching between global and local states of a system are introduced to change network structures. The categorical approach (Sch93) uses graph transformation on the network level and the local level. Distributed graph transformation as presented in this article combines structured graph transformation on two levels introduced in the categorical approach with the synchronization possibilities via interface graphs of the algebraic approach. Modeling of distributed systems by distributed graph transformation supports a clear and elegant description of dynamic networks, local actions, network administration, communication and synchronization of system components. Sample applications of distributed graph transformation like a revision management for distributed software engineering and distributed database transaction modeling can be found in (Tae96) and (Koc97).

Distributed graph transformation is formalized on the basis of the algebraic approach to graph transformation. A distributed graph is described as a diagram of graphs and graph morphisms. A transformation step is characterized by a double-pushout in the category **DISTR(GRAPH)** which consists of distributed graphs as objects and distributed graph morphisms. Category **DISTR(GRAPH)** differs from a normal functor category since the shape of diagrams may change during a transformation step. This feature allows the modeling of dynamic network structures. Split and join facilities as well as parallel transformations of local graphs are expressed by non-injective network morphisms. A distributed transformation step is based on the pushout construction in category **DISTR(GRAPH)**. To model the distributed behavior adequately, this pushout construction should be componentwise. The necessary and sufficient conditions for componentwise pushout construction are given in this article.

It is shown that distributed graph transformation fits into the high-level replacement framework in (EHKP91; EL93). High-level replacement systems are an axiomatic framework based on double-pushouts in order to unify replacement systems like grammars for different kinds of graphs and relational structures or other types of structures like algebraic specifications. It can be expected that results for high-level replacement systems concerning parallelism, concurrency and embedding can be carried over to distributed graph transformation.

The paper is organized as follows: As a basis for distributed graph transformation the original double-pushout approach to graph transformation is reviewed in section 2. Following this approach, graph nodes and edges may be not only glued but also split during a transformation step. Section 3 introduces distributed graphs and graph operations, especially the pushout construction in the category **DISTR(GRAPH)**.

Distributed graph transformation as presented in section 4 is proposed to describe the dynamic topology of a system as well as local and communicating actions. The conceptual ideas presented in this section lead to more elaborated and structured descriptions of essential communication concepts compared with those of existing graph transformation models for distributed systems. Distributed graph transformation is formally defined based on the double-pushout approach to graph transformation leading to a cautious rewriting style.

In the following, nearly all proofs are proof sketches. The full proofs can be found in (Tae96) and (Tae97). The reader is supposed to be familiar with the basic notions of category theory, especially colimits and functor categories as presented in e.g. (Mac71) or (HS73).

2. The Algebraic Approach to Graph Transformation

In this section, we review the basic ideas of the original algebraic approach to graph transformation, more precisely the double-pushout (DPO) approach which has been introduced in (EPS73). Comprehensive introductions to some restricted version are given in (Ehr79; CMR+96).

Definition 2.1 (Graphs and graph morphisms).

1. Let $C = (C^E, C^N)$ be a pair of color alphabets for nodes and edges. A *graph* $G = (G^E, G^N, s_G, t_G, m_{GE}, m_{GN})$ consists of the sets of edges and nodes, G^E and G^N, mappings $s_G, t_G : G^E \to G^N$, called source resp. target map, and the coloring maps for arcs and nodes, $m_{GE} : G^E \to C^E$ and $m_{GN} : G^N \to C^N$. G is said to be *colored according to* C.

2. Given two graphs G and H a *graph morphism* $f : G \to H$ is a pair of mappings $(f^E : G^E \to H^E, f^N : G^N \to H^N)$ such that f^E and f^N are compatible with source and target mappings and label preserving, i.e. $f^E(s_G(e)) = s_H(f^N(e))$, $f^E(t_G(e)) = t_H(f^N(e))$, $f^E(m_{GE}(e)) = m_{HE}(e)$ and $f^N(m_{GN}(n)) = m_{HN}(n)$, $\forall e \in G^E$ and $\forall n \in G^N$.

3. A graph morphism $f = (f^E, f^N)$ is called *injective (surjective, bijective)* if f^E and f^N are injective (surjective, bijective).

Remark: If we do not want to distinguish between nodes and edges the addition "N" or "E" is left out, e.g. "G" instead of "G^N" and "G^E". Most of the following operations on graphs are componentwise defined such as the disjoint union $G + H$ of graphs G and H consisting of the disjoint union of node and edge sets $G^N + H^N$ and $G^E + H^E$. If the constructions are completely analogous we do not distinguish the node and the edge case.

In the following examples, graphs and graph morphisms are given in a graphical way where the nodes and edges are not explicitly named. A set-theoretical description can be uniquely derived from the drawings. The mappings of nodes and edges are given by dashed edges between corresponding nodes. Since we only consider total morphisms the mapping of the edges is then uniquely determined up to isomorphism.

Definition 2.2 (Gluing of graphs). Given three graphs A, B and C and two graph morphisms $f : A \to B$ and $g : A \to C$ graph D and two graph morphisms $h : B \to D$ and $k : C \to D$ are constructed in the following way:

Let $\bar{I}_A \subseteq B \times C$ be the equivalence relation generated by the relation $I_A = \{(f(x), g(x)) \mid x \in A\}$. The *glued graph* D is then defined by $D = (B + C)_{/\bar{I}_A}$, the quotient of the disjoint union of B and C. Functions $h : B \to D$ and $k : C \to D$ send each element of B and C to its equivalence class in D.

Mappings s_D, t_D, m_{DN} and m_{GE} are uniquely determined by definition 2.2.

The following fact shows that all the essential notions of graph, graph morphism and gluing of graphs can be characterized by concepts from category theory.

Fact 2.3 (Category **GRAPH***).*

1. All graphs and graph morphisms as defined above form a category **GRAPH**.
2. Bijective graph morphisms are isomorphisms in **GRAPH**.
3. The empty graph $\emptyset$, i.e. a graph with empty node and edge sets, is the initial object in **GRAPH**.
4. The disjoint union $\sum_1^n G_i$ of graphs G_i, $1 \leq i \leq n$, with the injections $in_i : G_i \to \sum_1^n G_i$ is the coproduct of all G_i's in **GRAPH**.
5. The gluing construction in 2.2 can be characterized as a pushout in the category **GRAPH**.

Proof: Compare (EPS73; Ehr79).

□

Now, we start to describe sequential graph transformation which is performed by applying one production to the given graph. A production describes which nodes and edges shall be deleted, preserved and newly created. This is expressed by two graphs L and R, the left-hand side and the right-hand side. Moreover, we have an intermediate graph I, also called the gluing graph. This graph shows the preserved items. Their images in L and R correspond to the preserved items. Moreover, preserved items can be split or glued which is handled by the morphisms to the left- and right-hand sides that may be non-injective.

Definition 2.4 (Production). A *production* $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ consists of graphs L, I and R, called *left-hand side*, *intermediate* or *gluing graph* and *right-hand side* of p, respectively, and two morphisms $I \xrightarrow{l} L$ and $I \xrightarrow{r} R$.

- p is called *l-injective* (*l-surjective*) if morphism l is injective (surjective).
- p is called *injective* if morphisms l and r are injective.
- A production $p = (I \xleftarrow{id_I} I \xrightarrow{id_I} I)$ of two identities is called *identical production*.

Given a production a match of its left-hand side has to be searched in the host graph where the production should be applied. If nodes should be deleted or split we have to make sure that there is not any context edge which would dangle after deletion, i.e. which would not have a source or a target node afterwards. If a node with a context edge is split it is not clear which one of the new nodes should be equipped with the edge. Furthermore, the match need not to be an isomorphic image of the left-hand side. But we have to ensure that there are not two items identified where one is deleted and the other is preserved. Moreover, we should be careful with deletion and splitting which means isomorphic images of the deleting and the splitting part are demanded. All these restrictions are combined in the following gluing condition.

Definition 2.5 (Extended gluing condition). Given a production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$, a graph morphism $m : L \to G$ satisfies the *extended gluing condition* with respect to p if

1. Extended Identification-Condition:
   $\forall x_1, x_2 \in L$ with $m(x_1) = m(x_2)$ there is a unique $y_1 \in I$ and a unique $y_2 \in I$ with $l(y_1) = x_1$ and $l(y_2) = x_2$
2. Extended Dangling Condition:
   $\forall e \in G^E - m^E(L^E)$ let $L^N_e = \{x \in L^N \mid s_G(e) = m^N(x) \text{ or } t_G(e) = m^N(x)\}$ and $\forall x \in L^N_e$ there is a unique $y \in I^N$ with $l^N(y) = x$

Morphism m is called *match* if it satisfies the extended gluing condition wrt. p.

If morphism l is injective the identification condition and the dangling condition are no longer called extended. Identification and dangling condition form the gluing condition then.

If the extended gluing condition is satisfied a production can be applied at the corresponding match. In the first step, all those items of working graph G are deleted which have preimages in the left-hand side but not in the interface graph. Moreover, graph parts are split according to the production leading to the context graph. This graph shows some kind of intermediate state of graph transformation where deletion and splitting are executed but creation and gluing still have to be done. Afterwards the resulting graph is constructed.

Definition 2.6 (Context graph). Given a production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ and a match $m : L \to G$ which satisfies the extended gluing condition.

Let $SPLIT(p) = \{x \in I \mid \exists x' \neq x \in I : l(x) = l(x')\}$ then the *context graph* C is defined by $C = G - m(L - l(I)) - m(l(SPLIT(p))) + SPLIT(p)$. Mappings s_C, t_C, m_{CN} and m_{CE} are defined compatible with the definition of C^N and C^E.

The step of deletion and splitting can be described as inverse gluing of graphs. Taking the left-hand side L and the context graph C which overlap in the interface graph I the gluing of L and C along I yields graph G again. This is shown as follows. Applying an l-injective production to a graph the context graph can be characterized by a pushout complement (PO-complement). Furthermore, applying an l-surjective production to a graph this can be characterized by a so-called initial PO-complement. Since each graph morphism can be decomposed into a surjective followed by an injective one this can be done for each left-hand morphism of a production. The constructions over both parts are described by pushouts thus the whole deletion and splitting step is modeled by a pushout.

Fact 2.7 (Characterization of context graph - 1). Given an l-injective production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ and a match $m : L \to G$ there are a context graph C and two graph morphisms $g : C \to G$ as well as $c : I \to C$ such that (G, m, g) is the pushout of l and c in **GRAPH**. (C, c, g) is also called pushout complement of (G, m, l).

Proof: In this special case $SPLIT(p)$ is empty. Thus, context graph C looks like $C = G - m(L - l(I))$. Morphisms c and g are defined by $c = m \circ l$ and $g = id_{G/C}$. Compare (Ehr79) for the proof. □

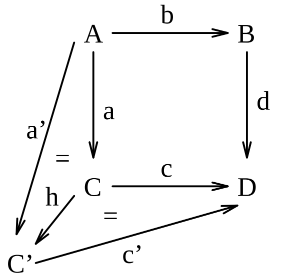

Figure 1. Initial pushout complement

Definition 2.8 (Initial PO-Complement). Given morphisms $b : A \to B$ and $d : B \to D$ a PO-complement $(C, a : A \to C, c : C \to D)$ is called *initial* if for all PO-complements $(C', a' : A \to C', c' : C' \to D)$ over b and d there is a unique morphism $h : C \to C'$ with $h \circ a = a'$ and $c' \circ h = c$. (Compare figure 2.8.)

Fact 2.9 (Uniqueness of initial PO-complements). The initial PO-complement $(C, a : A \to C, c : C \to D)$ over morphisms $b : A \to B$ and $d : B \to D$ is unique up to isomorphism.

Proof: Assume there are two initial PO-complements $(C, a : A \to C, c : C \to D)$ and $(C', a' : A \to C', c' : C' \to D)$. Then, there are morphisms $h : C \to C'$ with $h \circ a = a'$ and $c \circ h = c'$ and $h' : C' \to C$ with $h' \circ a' = a$ and $c' \circ h' = c$. Moreover, there is a morphism $h' \circ h$ with $c \circ h' \circ h = c' \circ h = c$ and $h' \circ h \circ a = h' \circ a' = a$. Since there is also $id_C : C \to C$ with $c \circ id_C = c$ and $id_C \circ a = a$ we have $h' \circ h = id_C$. Analogously, we obtain $h \circ h' = id_{C'}$. Hence, C and C' are isomorphic.

□

Proposition 2.10 (Characterization of context graph - 2). Given an l-surjective production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ and a match $m : L \to G$ there are a context graph C and two graph morphisms $g : C \to G$ as well as $c : I \to C$ such that (C, c, g) is the initial PO-complement over l and m in **GRAPH**.

Proof sketch: In this special case $m(L - l(I))$ is empty. Thus, context graph C looks like $C = G - m(l(SPLIT(p))) + SPLIT(p)$. C is a graph since s_C, t_C, m_{CN} and m_{CE} are well-defined mappings. Morphisms c and g are defined as follows.

$$c(x) := \begin{cases} m(l(x)), & \text{if } x \notin SPLIT(p) \\ x, & \text{if } x \in SPLIT(p) \end{cases}$$

$$g(x) := \begin{cases} m(l(x)), & \text{if } x \in SPLIT(p) \\ x, & \text{if } x \notin SPLIT(p) \end{cases}$$

c and g are graph morphisms since c and g are well-defined mappings and compatibility with source, target and coloring maps can be shown.

It remains to prove the pushout properties. The commutativity: $m \circ l = g \circ c$ is straight forward to show. If there is a graph X and two morphisms $m' : L \to X$ and $g' : C \to X$ with $m' \circ l = g' \circ c$ we have to show that there is a unique morphism $h : G \to X$ with $h \circ m = m'$ and $h \circ g = g'$. Let h be defined by

$$h(x) := \begin{cases} m'(m^{-1}(x)), & \text{if } x \in m(L - l(I))) \\ g'(g^{-1}(x)), & \text{if } x \in g(C) \end{cases}$$

The well-definedness of h can be shown on the basis of the commutativity property above. It is easy to show that $h \circ m = m'$ and $h \circ g = g'$ and that h is the only morphism with this property.

$$\begin{tikzcd}
 & A \arrow[r, "b"] \arrow[d, "a"] \arrow[ddl, "a'"'] \arrow[dr, phantom, "(1)"] & B \arrow[r, "e"] \arrow[d, "d"] \arrow[dr, phantom, "(2)"] & E \arrow[d, "f"] \\
 & C \arrow[r, "c"] \arrow[dl, "h"] & D \arrow[r, "g"] & F \\
C' \arrow[urr, "c'"] \arrow[urrr, "g'"'] & & &
\end{tikzcd}$$

Figure 2. Pushout Composition

Next we have to show that (C, c, g) is initial which means that for each PO-complement $(C', c' : K \to C', g' : C' \to G)$ over m and l there is a unique graph morphism $j : C \to C'$ with $j \circ c = c'$ and $g' \circ j = g$. Let j be defined by

$$j(x) = y \text{ if } g(x) = g'(y) \wedge \forall z \in I \text{ with } c(z) = x \text{ implies } c'(z) = y$$

j is a graph morphism and $g' \circ j = g$ and $j \circ c = c'$ can be shown directly from the definition of j. Finally, the uniqueness of j is also easy to show. □

Proposition 2.11 (Pushout properties with initial PO-complements). Given the diagram in figure 2.11 where (1) and (2) are pushouts, morphism e is injective and (C, a, c) is an initial PO-complement of b and d, then also $(C, a, g \circ c)$ is an initial PO-complement in **GRAPH**.

Proof: Diagram (1+2) is a pushout according to general pushout properties. Given a PO-complement (C', a', g') as depicted in figure 2.11 we have to show that there is a unique morphism $h : C \to C'$ with $h \circ a = a'$ and $g' \circ h = c \circ g$. According to assumptions $g' \circ a' = f \circ e \circ b = g \circ c \circ a$ holds. We can define $c' = g^{-1} \circ g'$. g^{-1} is defined for all $g'(x)$ with $x \in C'$ since g and f are jointly surjective according to pushout (2). For pushouts in **GRAPH** we have the injectivity of g if e is injective. (Compare pushout properties for graphs in (Wag93).) Thus, c' is well-defined. According to initial PO-complement (C, a, c) we have a unique morphism h with $h \circ a = a'$ and $c' \circ h = c$. With $g \circ c' = g'$ we get $g' \circ h = g \circ c' \circ h = g \circ c$. Thus, $(C, a, g \circ c)$ is initial.

□

Proposition 2.12 (Applicability of production). Given a production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ and a match $m : L \to G$ there are a context graph C and two total graph morphisms $g : C \to G$ as well as $c : I \to C$ which form the initial PO-complement of m and l.

Proof: According to fact 2.3 morphism l can be decomposed into a surjective morphism $l_e : I \to I_{/\equiv}$ and an injective morphism $l_i : I_{/\equiv} \to L$. Looking at morphisms m and l_i there is a PO-complement $(\bar{C}, g_i : \bar{C} \to G, \bar{c} : I_{/\equiv} \to \bar{C})$ using fact 2.7. Since m satisfies the extended gluing condition and $\bar{c}$ is nearly a restriction of m it satisfies the extended gluing condition, too. Furthermore, there is an initial PO-complement $(C, c : I \to C, g_e : C \to \bar{C})$ according to proposition 2.10. Let $g = g_i \circ g_e$. Using proposition 2.11 there is an initial PO-complement (C, c, g). (Compare figure 2.)

□

For $\bar{C}$ it can be shown that it is the only PO-complement graph. But C is not the only possible PO-complement graph. It describes that one where most is split. All other PO-complement graphs are factorizations of C in some sense. Thus, we will find a unique morphism from C to every other PO-complement graph which is described by the additional property of a PO-complement to be initial. Hence, the whole construction of C can be proven to be unique up to isomorphism.

Proposition 2.13 (Characterization of (extended) gluing condition). If $(C, g : C \to G, c : I \to C)$ is an (initial) PO-complement of $l : I \to L$ and $m : L \to G$ then m satisfies the (extended) gluing condition wrt. $p = (L \xleftarrow{l} I \xrightarrow{r} R)$.

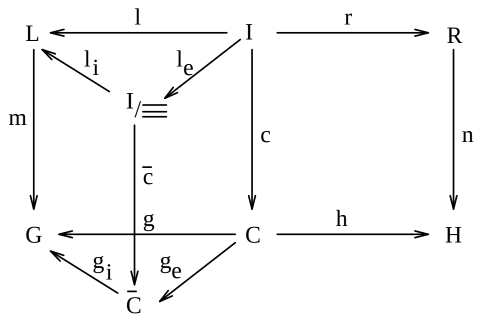

Figure 3. DPO-graph transformation with splitting

$$\begin{tikzcd} L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & I \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "c"] \arrow[dr, phantom, "(2)"] & R \arrow[d, "q"] \\ G & C \arrow[l, "c_g"] \arrow[r, "c_h"'] & H \end{tikzcd}$$

Figure 4. Direct transformation

Proof sketch: If (C, g, c) is a PO-complement of m and l m satisfies the gluing condition wrt. p (compare (Ehr79)). If(C, g, c) is moreover initial we can show the extended gluing condition (compare (Tae97)). □

Definition 2.14 (Direct transformation). Given a production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ and a match $m : L \to G$, a *direct transformation* $G \Longrightarrow_d H$ using p and m, short $G \overset{p,m}{\Longrightarrow}_d H$, from a graph G to a graph H is given by the two pushout diagrams (1) and (2) in the category **GRAPH** (shown in figure 2.14) where (C, c, c_g) is an initial PO-complement.

Proposition 2.15 (Uniqueness of direct transformations). Given a production $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ and a match $m : L \to G$ the direct transformation $G \overset{p,m}{\Longrightarrow}_d H$ is unique up to isomorphism.

Proof: Due to proposition 2.12 and the fact that pushouts are unique up to isomorphism. □

3. The Category of Distributed Graphs

Distributed states can be described by distributed graphs introduced below. To be able to describe distributed graph transformation we introduce a suitable kind of morphisms, the distributed graph morphisms. Furthermore, we need the construction of a pushout of distributed graph morphisms to characterize gluing facilities for distributed graphs. In this section, conditions are given to ensure the existence of a componentwise construction of such a pushout which reflects a distributed procedure best. The conditions for componentwise construction are sufficient and necessary.

Definition 3.1 (D-graph). Let **GRAPH** be the category of labeled graphs and (total) graph morphisms which are label preserving (as presented in fact 2.3). Given a graph G of **GRAPH** which is called *network graph* a diagram $\hat{G} : G \to$ **GRAPH** is called *distributed graph (D-graph)*[1]. D-graph $\hat{G}$ is said to *be of shape* G.

A local graph is denoted by $\hat{G}(i)$ for any network node $i \in G^N$. Given a network edge $e \in G^E$, the local graph $\hat{G}(s(e))$[2] is an interface graph or source graph, $\hat{G}(t(e))$ is called target graph and $\hat{G}(e)$ is a total graph morphism.

[1] Graph G and its induced small category are identified.

[2] In the following, it is always clear which source or target mapping is meant. Thus, we do not write $\hat{G}(s_G(e))$ $(\hat{G}(t_G(e)))$ but simpler $\hat{G}(s(e))$ $(\hat{G}(t(e)))$.

$$
\begin{tikzcd}
i \arrow[d, "e"'] & \hat{G}(i) \arrow[r, "\hat{f}_i"] \arrow[d, "\hat{G}(e)"'] \arrow[dr, phantom, "(=)"] & \hat{H}(f(i)) \arrow[d, "\hat{H}(f(e))"] \\
j & \hat{G}(j) \arrow[r, "\hat{f}_j"'] & \hat{H}(f(j))
\end{tikzcd}
$$

Figure 5. D-graph morphism

$$
\begin{tikzcd}
i \arrow[d, "e"'] & \hat{G}(i) \arrow[r, "\hat{g}_i"] \arrow[d, "\hat{G}(e)"'] & \hat{K}(g(i)) \arrow[r, "\hat{h}_{g(i)}"] \arrow[d, "\hat{K}(g(e))"'] & \hat{H}(h(g(i))) \arrow[d, "\hat{H}(h(g(e)))"] \\
j & \hat{G}(j) \arrow[r, "\hat{g}_j"] & \hat{K}(g(j)) \arrow[r, "\hat{h}_{g(j)}"] & \hat{H}(h(g(j)))
\end{tikzcd}
$$

Figure 6. Composition of D-graph morphisms

Definition 3.2 (D-graph morphism). Given two D-graphs $\hat{G}$ and $\hat{H}$ a distributed graph morphism, short *D-graph morphism* $\hat{f} = (\eta, f) : \hat{G} \to \hat{H}$ is a natural transformation $\eta = (\hat{f}_i)_{i \in G^N} : \hat{G} \to \hat{H} \circ f$ in **GRAPH** with $f : G \to H$ being a graph morphism of **GRAPH**, called *network morphism.*

If f is injective (surjective) $\hat{f}$ is called *n-injective* (*n-surjective*). If moreover, all $\hat{f}_i$ with $i \in G^N$ are injective, $\hat{f}$ is called *injective*, too.

A D-graph morphism $\hat{f}$ assigns to each node $i \in G^N$ a graph morphism $\hat{f}_i : \hat{G}(i) \to \hat{H}(f(i))$, called **local graph morphism** such that $\forall e \in G^E$ where $s(e) = i$ and $t(e) = j$ the diagram in figure 3 commutes. More concretely, given two local graphs $\hat{G}(i)$ and $\hat{G}(j)$ which are connected by $\hat{G}(e)$ and these graphs are mapped by local graph morphisms $\hat{f}_i$ and $\hat{f}_j$ to graphs $\hat{H}(f(i))$ and $\hat{H}(f(j))$ the connection $\hat{G}(e)$ has to be adapted leading to $\hat{H}(f(e))$.

Definition 3.3 (Composition of D-graph morphisms). The *composition* $\hat{f} = (\eta_f, f) : \hat{G} \to \hat{H}$ *of two D-graph morphisms* $\hat{g} = (\eta_g, g) : \hat{G} \to \hat{K}$ and $\hat{h} = (\eta_h, h) : \hat{K} \to \hat{H}$ is defined by $f = h \circ g$ and $\eta_f = (\eta_h \circ g) \circ \eta_g : \hat{G} \to \hat{K} \circ g \to \hat{H} \circ h \circ g$.

Proposition 3.4 (Category **DISTR(GRAPH)***).* All D-graphs and D-graph morphisms as defined above form a category **DISTR(GRAPH)**.

Proof: The composition $\hat{f} : \hat{G} \to \hat{H}$ of two D-graph morphisms $\hat{g} : \hat{G} \to \hat{K}$ and $\hat{h} : \hat{K} \to \hat{H}$ is again a D-graph morphism. According to definition 3.3 we have $\hat{f}_i = \hat{h}_{g(i)} \circ \hat{g}_i$ for all $i \in G^N$, and for all $e \in G^E$ with $s_G(e) = i$ and $t_G(e) = j$ the diagrams in figure 3 commute. Since the composition of D-graph morphisms is obtained by that of graph morphisms it is associative.

The identities $\hat{id}_{\hat{G}} = (\eta_{id_G}, id_G) : \hat{G} \to \hat{G}$ for each D-graph $\hat{G}$ are provided by the identities $\hat{id}_{\hat{G}(i)} : \hat{G}(i) \to \hat{G}(i)$ for all $i \in G^N$. $\hat{id}_{\hat{G}} \circ \hat{g} = \hat{g}$ and $\hat{f} \circ \hat{id}_{\hat{G}} = \hat{f}$ is satisfied for all D-graph morphisms $\hat{f} = (\eta_f, f) : \hat{G} \to \hat{H}$ and $\hat{g} = (\eta_g, g) : \hat{H} \to \hat{G}$, since $id_G \circ g = g$ and $\eta_g = (\eta_{id_G} \circ g) \circ \eta_g : \hat{H} \to \hat{G} \circ g \to \hat{G} \circ g$. Moreover, $f \circ id_G = f$ and $\eta_f = \eta_f \circ \eta_{id_G} : \hat{G} \to \hat{G} \to \hat{H} \circ f$.

$\square$

Proposition 3.5 (Colimit of a D-graph). Let $\hat{G}$ be a D-graph, $\sum \hat{G}(n)$ is the coproduct of all $\hat{G}(n)$, $n \in G^N$, $i_n : \hat{G}(n) \to \sum \hat{G}(n)$are the corresponding injections. Let $\equiv_G$ be the equivalence closure of $\{(i_{s(e)}(x), i_{t(e)}(\hat{G}(e)(x))) | x \in \hat{G}(s(e)), e \in G^E\}$. Then, $\hat{G}' = \sum \hat{G}(n)_{/\equiv_G}$ and

$\hat{g}' : \sum \hat{G}(n) \to \hat{G}'$ is the corresponding quotient function. Morphisms $\hat{g}' \circ i_n : \hat{G}(n) \to \hat{G}\ \forall n \in G^N$ form the colimit of $\hat{G}$.

Proof: Compare (Wol95) for the colimit construction of an arbitrary diagram in the category **SET** of sets and functions. This construction can be directly lifted to the category **GRAPH** which can be described as comma category over **SET**. □

Definition 3.6 (Equivalence relations). Given a D-graph morphism $\hat{a} : \hat{A} \to \hat{B}$ let $\sim_a = \{(x, y) \in A \times A | a(x) = a(y)\}$ be the *kernel* of a. $equiv(\sim_a)$ denotes the smallest equivalence relation induced by $\sim_a$.

Given an equivalence relation $\approx$ let $\hat{A}_{[x]}$ be a distributed graph which consists of all graphs $\hat{A}(y)$ for $y \in [x]$ and all graph morphisms $\hat{A}(e)$ for $s(e), t(e) \in [x]$ for each class $[x] \in A^N_{/\approx}$. $\hat{A}_{[x]}$ is called *gluing section* $[x]$ *of* $\hat{A}$.

The restriction of an equivalence relation $\approx$ to nodes (edges) is depicted by $\approx^N$ ($\approx^E$). Let $a(\sim) = \{(a(x), a(y)) \in B \times B | (x, y) \in \sim\}$.

Proposition 3.7 (E-M-Factorization). Given a D-graph morphism $\hat{f} : \hat{G} \to \hat{H}$, $\hat{f}$ can be decomposed into an n-surjective D-graph morphism $\hat{g} : \hat{G} \to \hat{K}$ where $\forall [x] \in G^N_{/\approx_f}$ $\forall \hat{g}_y$ with $y \in [x]$ are jointly surjective and an injective D-graph morphism $\hat{h} : \hat{K} \to \hat{H}$ such that $\hat{f} = \hat{h} \circ \hat{g}$. Moreover, $\hat{K}$ is unique up to isomorphism. $\hat{g}$ is an epimorphism and $\hat{h}$ is a monomorphism.

Proof sketch: Graph morphism f can be decomposed into a surjective graph morphism g and an injective graph morphism h with $f = h \circ g$ such that K is unique up to isomorphism. This is due to fact 2.3.

$\forall [x] \in G^N_{/\approx_f}$ let $(C_{[x]}, \bigcup_{y \in [x]} c_y : \hat{G}(y) \to C_{[x]})$ be the coproduct. Furthermore, $\forall [x] \in G^N_{/\approx_f}$ $u_{[x]} : C_{[x]} \to \hat{H}(f(x))$ are the induced coproduct morphisms. Each of them can be factorized into a surjective graph morphism $s_{[x]} : C_{[x]} \to \hat{K}(g(x))$ and an injective graph morphism $h_{[x]} : \hat{K}(g(x)) \to \hat{H}(f(x))$. $\forall e \in G^E$ $\hat{K}(g(e))$ is defined by $h^{-1}_{[t(e)]} \circ \hat{H}(f(e)) \circ h_{[s(e)]}$. $\hat{g}_x = s_{[x]} \circ c_x\ \forall x \in G^N$. It is left to show that $\hat{g}$ and $\hat{h}$ are D-graph morphisms and $\hat{K}$ is unique up to isomorphism. Moreover, $\hat{g}$ and $\hat{h}$ have to be shown to be epi- and monomorphism, resp. The full proofs can be found in (Koc97). □

In the following, we do not consider the existence of pushouts in the category **DISTR(GRAPH)** in general, but concentrate on those which can be constructed componentwise. First, we concentrate on the pushout construction over n-injective morphisms.

A pushout over n-injective D-graph morphisms $\hat{a} : \hat{A} \to \hat{C}$ and $\hat{b} : \hat{A} \to \hat{B}$ can be constructed componentwise in the following steps. First the pushout on network morphisms is created. For all network nodes in A the pushout on their local graph morphisms into $\hat{B}$ and $\hat{C}$ is constructed. All other local graphs of $\hat{B}$ and $\hat{C}$ are carried over to $\hat{D}$ unchanged. A network edge in D which has a preimage in A is equipped with the induced morphism between its source and target pushout graph. All other local graph morphisms of $\hat{B}$ and $\hat{C}$ are adapted to their new target graphs. The source graphs have to be mapped structure equivalent according to the pushout conditions.

One might argue that a gluing of distributed graphs where first the gluing of the corresponding network graphs is computed is not distributed. But this is just a compact description of what has to be done. The gluing of graphs especially with injective morphisms in between can be done elementwise. That means for example, a node in B and one in C which have a common origin in A are glued. All others are just copied to D. These actions can be done in parallel.

The componentwise construction of pushouts over n-injective morphisms can always be performed if the following two pushout conditions are satisfied. These conditions require for each network edge which has a source network node that is glued with another, that the local graph of this network node is isomorphically mapped into the pushout graph.

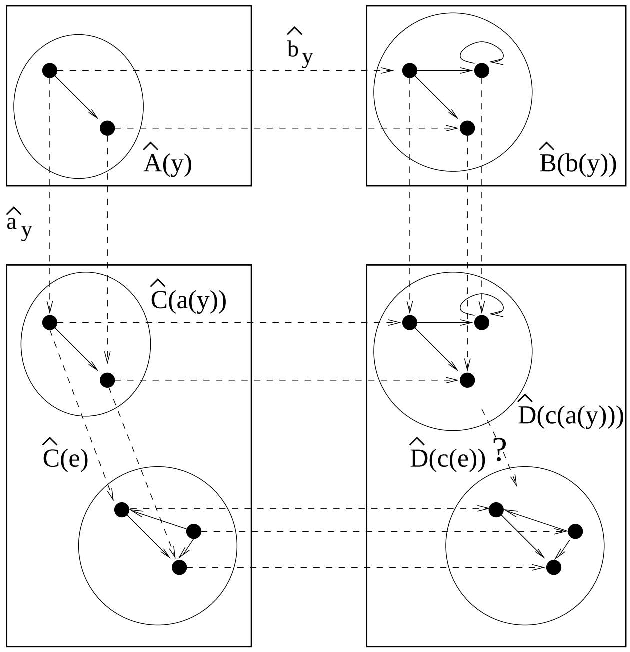

Figure 7. A gluing situation not allowed

Definition 3.8 (Pushout Conditions (1) and (2)). Two n-injective D-graph morphisms $\hat{a} : \hat{A} \to \hat{C}$ and $\hat{b} : \hat{A} \to \hat{B}$ satisfy the *pushout conditions* (1) and (2) if
(1) $\forall e \in C^E - a(A^E) \forall y \in A^N : a(y) = s(e) \Longrightarrow \hat{b}_y$ is surjective and $\sim_{\hat{b}_y} \subseteq \sim_{\hat{a}_y}$
(2) $\forall e \in B^E - b(A^E) \forall y \in A^N : b(y) = s(e) \Longrightarrow \hat{a}_y$ is surjective and $\sim_{\hat{a}_y} \subseteq \sim_{\hat{b}_y}$.

Example 3.9 (Pushout Conditions (1) and (2)). In figure 3.9 a situation is shown where pushout condition (1) is not satisfied. All local graph morphisms are depicted by dashed arrows between corresponding nodes. The mapping of edges can be uniquely derived. The network morphisms are not explicitly shown.

Morphism $\hat{b}_y$ is not surjective. The node with a loop and its adjacent edges in $\hat{B}(b(y))$ which are copied into $\hat{D}(c(a(y)))$ cannot be mapped by $\hat{D}(c(e))$.

Proposition 3.10 (Existence of Pushouts). Given two n-injective D-graph morphisms $\hat{a} : \hat{A} \to \hat{C}$ and $\hat{b} : \hat{A} \to \hat{B}$ which satisfy pushout conditions (1) and (2) the pushout of $\hat{a}$ and $\hat{b}$ in **DISTR**(**GRAPH**) exists and can be constructed componentwise.

Proof sketch: Let D with $c : C \to D$ and $d : B \to D$ be the pushout of a and b in **GRAPH**. (For the construction compare definition 2.2.) Pushout graph $\hat{D} : D \to \mathbf{GRAPH}$ is constructed in the following way:

$\forall x \in D^N$ there is at most one $y \in A^N$ with $c \circ a(y) = x$ because a and b are injective. Thus we can uniquely name $PO(\hat{a}_y, \hat{b}_y)$[3] by PO_x.

$$\hat{D}(x) := \begin{cases} PO_x & \text{, if } \exists y \in A^N \text{ with } c \circ a(y) = x \\ \hat{C}(z) & \text{, if } \exists z \in C^N - a(A^N) \text{ with } c(z) = x \\ \hat{B}(v) & \text{, if } \exists v \in B^N - b(A^N) \text{ with } d(v) = x \\ IND(PO_{s(x)}, PO_{t(x)}) & \text{, if } \exists e \in A^E \text{ with } c \circ a(e) = x \\ \hat{c}_{t(x)} \circ \hat{C}(e) \circ \hat{c}_{s(x)}^{-1} & \text{, if } \exists e \in C^E - a(A^E) \text{ with } c(e) = x \\ \hat{d}_{t(x)} \circ \hat{B}(e) \circ \hat{d}_{s(x)}^{-1} & \text{, if } \exists e \in B^E - b(A^E) \text{ with } d(e) = x \end{cases}$$

[3] $PO(\hat{a}_y, \hat{b}_y)$ is the pushout graph of the pushout over $\hat{a}_y$ and $\hat{b}_y$

where $IND(PO_{s(x)}, PO_{t(x)})$ is the induced morphism from pushout graph $\hat{D}(s(x))$ to pushout graph $\hat{D}(t(x))$ with $\hat{a}_{t(e)} \circ \hat{A}(e) = \hat{C}(a(e)) \circ \hat{a}_{s(e)}$ and $\hat{b}_{t(e)} \circ \hat{A}(e) = \hat{B}(b(e)) \circ \hat{b}_{s(e)}$. Pushout morphisms $\hat{c} : \hat{C} \to \hat{D}$ and $\hat{d} : \hat{B} \to \hat{D}$ are defined as follows:

$$\hat{c}_x := \begin{cases} id_{\hat{C}(x)} & \text{, if } x \in C^N - a(A^N) \\ \text{PO-morphism of } PO_{c(x)} & \text{, otherwise} \end{cases}$$

$$\hat{d}_x := \begin{cases} id_{\hat{B}(x)} & \text{, if } x \in B^N - b(A^N) \\ \text{PO-morphism of } PO_{d(x)} & \text{, otherwise} \end{cases}$$

It is straight forward to show that $\hat{D}$ is a D-graph using the above definition, PO-properties and the pushout conditions. It is next to show that $\hat{c}$ as well as $\hat{d}$ are D-graph morphisms using mainly the induced morphisms of local pushouts PO_x, $\forall x \in A^N$.

Now, the pushout properties have to be shown. The commutativity $\hat{c} \circ \hat{a} = \hat{d} \circ \hat{b}$ follows directly from the construction. Given a D-graph $\hat{X}$ as well as two D-graph morphisms $\hat{f} : \hat{B} \to \hat{X}$ and $\hat{g} : \hat{C} \to \hat{X}$ with $\hat{f} \circ \hat{b} = \hat{g} \circ \hat{a}$ the induced morphism $\hat{u}$ for the universal property consists of the induced morphisms for the underlying pushouts in **GRAPH** where they exist, i.e. for all $y \in A^N$ with $c \circ a(y) = x$, $\hat{u}_x$ is the induced morphism wrt. PO_x. For all $z \in C^N - a(A^N)$ with $c(z) = x$ ($v \in B^N - b(A^N)$ with $d(v) = x$), $\hat{u}_x = \hat{g}_z$ ($\hat{u}_x = \hat{f}_v$).

Well-definedness of $\hat{u}$ follows from the universal property of local pushouts and the pushout conditions. We have to show that $\hat{u}$ is totally defined which is the case since c and d are jointly surjective. Thus, there is a morphism u_x for all $x \in D^N$. Furthermore, $\hat{u}_j \circ \hat{D}(e) = \hat{X}(u(e)) \circ \hat{u}_i$ with $s(e) = i$ and $t(e) = j$ can be shown based on the universal property of the local pushouts and the pushout conditions.

Next we have to show that $\hat{u} \circ \hat{c} = \hat{g}$ holds. $u \circ c = g$ holds because of pushout properties of D. For all $x \in C^N$ we have $\hat{u}_{c(x)} \circ \hat{c}_x = \hat{g}_x$ according to the definition of $\hat{u}$. $\hat{u} \circ \hat{d} = \hat{f}$ can be proven analogously.

The uniqueness of $\hat{u}$ which means that for all $\hat{u}' : \hat{D} \to \hat{X}$ with $\hat{u}' \circ \hat{c} = \hat{g}$ and $\hat{u}' \circ \hat{d} = \hat{f}$ $\hat{u} = \hat{u}'$ holds, can be obtained directly from the definition of $\hat{u}$. Assuming there is a $y \in D^N$ such that $\hat{u}_y \neq \hat{u}'_y$ and there is an $x \in C^N$ with $c(x) = y$. Then, we have $\hat{u}_y \circ \hat{c}_x = \hat{g}_x = \hat{u}'_y \circ \hat{c}_x$. Thus, $\hat{u}$ is unique. □

Example 3.11 (Pushout in **DISTR(GRAPH)***).* Figure 3.11 shows an example of a pushout over two n-injective D-graph morphisms. The pushout conditions are satisfied since $\hat{a}_j : \hat{A}(j) \to \hat{C}(l)$ is bijective. Note that all other local graph morphisms need not to be injective or surjective. The local graph morphism between $\hat{D}(c(k))$ and $\hat{D}(c(l))$ is constructed as the induced morphism between both pushout graphs. Since $\hat{a}_j$ is bijective it is clear how the local graph morphism $\hat{D}(d(e'))$ has to be constructed. This local graph morphism is defined by $\hat{D}(d(e')) = \hat{d}_x \circ \hat{B}(e') \circ \hat{d}_w^{-1}$. If for example $\hat{C}(l)$ would contain an additional node, morphism $\hat{d}_w : \hat{B}(w) \to \hat{D}(c(l))$ would not be bijective and could not be inverted. Local graph morphism $\hat{D}(c(e))$ is constructed by $\hat{D}(c(e')) = \hat{c}_l \circ \hat{C}(e) \circ \hat{c}_m^{-1}$.

An example where the pushout cannot be constructed componentwise has been shown in example 3.9. The following proposition states that the pushout conditions (1) and (2) are not only sufficient but also necessary for componentwise pushout construction in category **DISTR(GRAPH)** for n-injective D-graph morphisms.

Proposition 3.12 (Componentwise PO-construction in **DISTR(GRAPH)**- *1).* Given two n-injective D-graph morphisms $\hat{a} : \hat{A} \to \hat{C}$ and $\hat{b} : \hat{A} \to \hat{B}$, the pushout of $\hat{a}$ and $\hat{b}$ in **DISTR**(**GRAPH**) can be constructed componentwise if and only if pushout conditions (1) and (2) are satisfied.

Proof: Proposition 3.10 states that pushout conditions (1) and (2) are sufficient for the componentwise construction of the pushout of $\hat{a}$ and $\hat{b}$. It remains to show that the pushout conditions are also necessary for the componentwise pushout construction. Assuming that pushout condition (1) is not

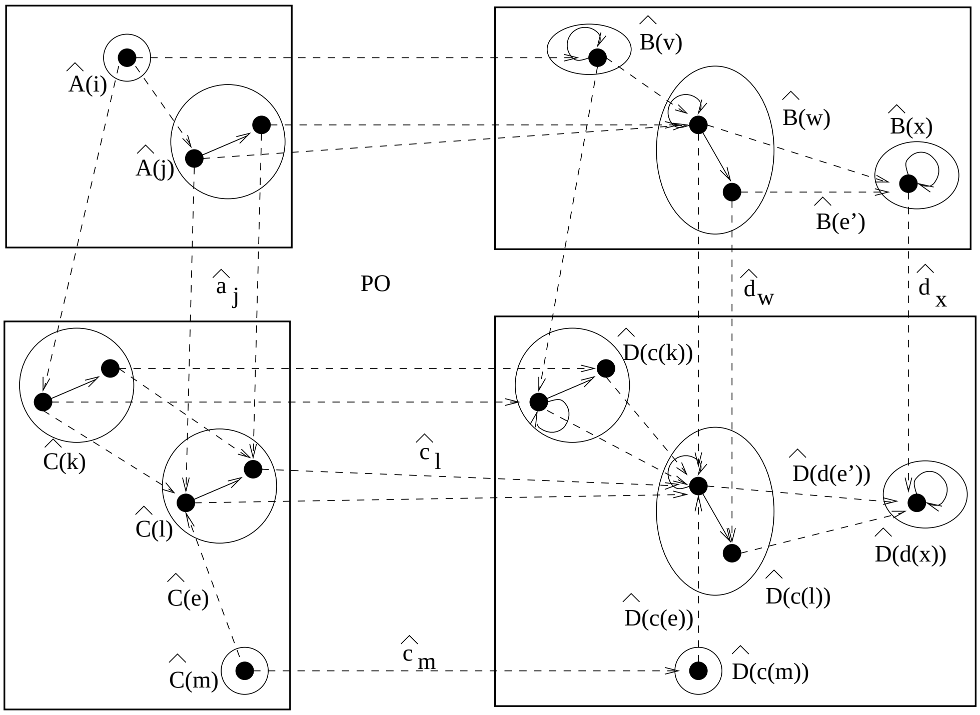

Figure 8. Example for a pushout in **DISTR(GRAPH)**

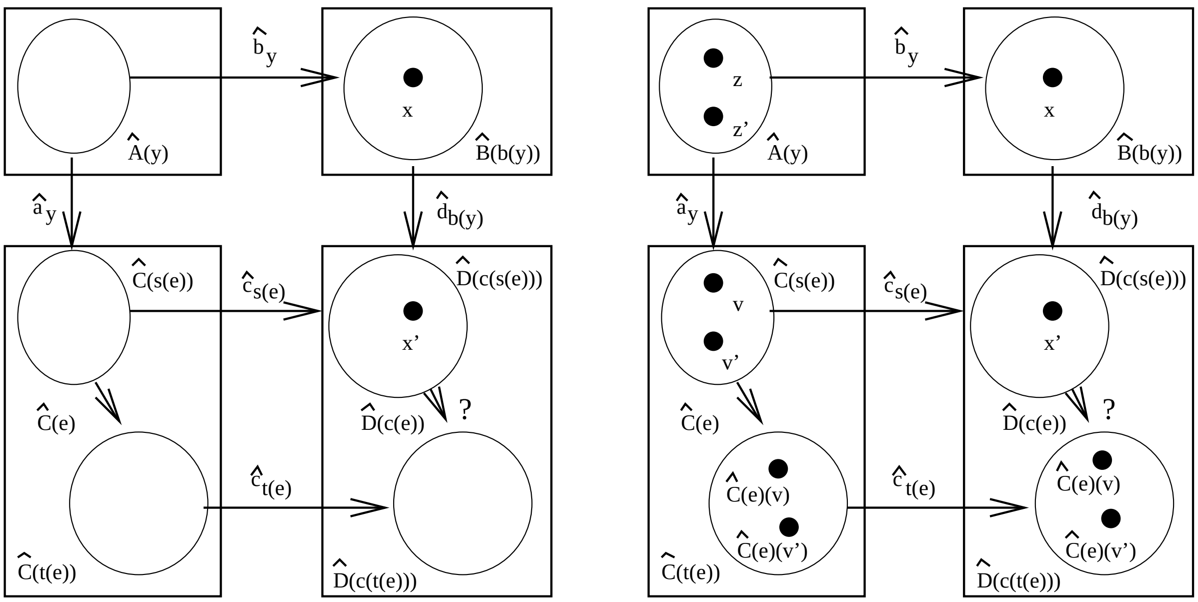

Figure 9. Illustrations for the proof of 3.12

satisfied there is a network edge $e \in C^E - a(A^E)$ such that there is a $y \in A^N$ with $a(y) = s(e)$ and $\hat{b}_y$ is not bijective. If $\hat{b}_y$ is not surjective there is a node $x \in \hat{B}(b(y))$ which does not have a preimage in $\hat{A}(y)$. (Compare the left diagram in figure 3 for this part of the proof.) Since $\hat{d}_{b(y)}$ has to be total there is also a node $x' \in \hat{D}(d(b(y))) = \hat{D}(c(s(e))$. Since $\hat{D}(c(s(e))$ is constructed as pushout graph of morphisms $\hat{a}_y$ and $\hat{b}_y$, local graph $\hat{C}(s(e))$ does not contain a preimage of x'. According to componentwise construction local graphs like $\hat{C}(t(e))$ which do not have a preimage in $\hat{A}$, are not changed, i.e. $\hat{D}(c(t(e))) = \hat{C}(t(e))$. Thus, it is possible to choose a local graph $\hat{C}(t(e)) = \hat{D}(c(t(e)))$ such that there is not a local graph morphism $\hat{D}(c(e)) : \hat{D}(c(s(e))) \to \hat{D}(c(t(e)))$.

If $\sim_{\hat{b}_y} \not\subseteq \sim_{\hat{a}_y}$ there is a node $x \in \hat{B}(b(y))$ which has two preimages $z, z' \in \hat{A}(y)$ with $z \neq z'$. Moreover, there are two nodes $v, v' \in \hat{C}(s(e))$ $\hat{a}_y(z) = v \neq v' = \hat{a}_y(z')$. (Compare the right diagram in figure 3 for this part of the proof.) According to pushout construction x has to be mapped to a node $x' \in \hat{D}(c(s(e)))$. If $\hat{C}(e)(v) \neq \hat{C}(e)(v')$, a local graph morphism $\hat{D}(c(e)) : \hat{D}(c(s(e))) \to \hat{D}(c(t(e)))$ can be chosen such that node $x' \in \hat{D}(c(s(e)))$ cannot be mapped to a node in $\hat{D}(c(t(e)))$. □

Now we go a step further and consider the gluing of distributed graphs connected by arbitrary distributed graph morphisms. This means that local graphs inside of one distributed graph can be glued. The construction of this advanced gluing operation is mainly done in two steps: first all the local gluing is done and then the (abstract) union of distributed graphs is performed which corresponds exactly to the pushout construction over n-injective D-graph morphisms. To perform the construction in this way which is again as modular as possible an additional pushout condition is needed which is stated in definition 3.14.

Definition 3.13 (Network gluing condition). Given an equivalence relation $\approx$ on a D-graph $\hat{D}$, $\hat{D}$ with $\approx$ satisfies the *network gluing condition* if $\forall x \in D^N$ with $x \in [s(e)]$ for some $e \in D^E$ and $\neg\exists e' \in [e]$ with $s(e') = x$: Let $\hat{D}_{[s(e)]}$ be a gluing section and $\hat{D}'([s(e)])$ the corresponding colimit graph:

$$\forall [y] \in \hat{D}'([s(e)]) \exists y' \in [y] : y' \in \hat{D}(s(e))$$

Definition 3.14 (Pushout Condition (3)). Given two D-graph morphisms $\hat{a} : \hat{A} \to \hat{C}$ and $\hat{b} : \hat{A} \to \hat{B}$ morphisms $\hat{a}$ and $\hat{b}$ satisfy the **pushout condition** (3) if

(a) $\hat{A}$ with $equiv(\sim_a \cup \sim_b)$ satisfies the network gluing condition.

(b) $\hat{B}$ with $equiv(b(\sim_a))$ satisfies the network gluing condition.

(c) $\hat{C}$ with $equiv(a(\sim_b))$ satisfies the network gluing condition.

Example 3.15 (Pushout Condition (3)). In figure 3.15 a situation is depicted where pushout condition (3) is not satisfied. Constructing the coproduct of $\hat{A}(x)$ and $\hat{A}(s(e))$ the resulting graph $\hat{A}'([s(e)])$ consists of two nodes. This is the colimit graph of $\hat{A}_{[s(e)]}$. Both node classes in $\hat{A}'([s(e)])$ contain only one node but the left one comes from $\hat{A}(x)$. Constructing $\hat{A}'$ it is not clear how graph morphism $\hat{A}(e)$ should look like. The only possibility to get a total graph morphism is to map both nodes of $\hat{A}'([s(e)])$ to the only node of $\hat{A}'([t(e)])$. But then, there is no way to get $\hat{a}'$ as a D-graph morphism assuming that $\hat{C}$ is isomorphic to $\hat{C}'$ if nothing is glued. A similar situation is shown for distributed graph $\hat{B}$.

The construction of a pushout over distributed graph morphisms which are not necessary n-injective leads back to that over n-injective ones. First all gluing the network morphisms demand is done. This means that all those local graphs in $\hat{A}$ are glued whose corresponding network nodes in A are in the same equivalence class induced by $\sim_{ab}$, i.e. the network node indicating that equivalence class is equipped with the corresponding gluing graph. Similar gluings of local graphs are performed in $\hat{B}$ and $\hat{C}$, too. Since the same equivalence relation $\sim_{ab}$ is used in $\hat{A}$, $\hat{B}$ and $\hat{C}$ there is a one-to-one relation between the corresponding classes and gluing graphs. Thus, $\hat{a}'$ and $\hat{b}'$ are n-injective and their local graph morphisms can be chosen as the induced ones according to the gluing graphs. The same ideas have been followed by gluing the network edges equipped with graph morphisms. The resulting edge class is equipped with the induced morphism according to the source gluing graph and all morphisms belonging to edges in that class. According to pushout condition (3) all classes in the source gluing graph have to contain an item in some source graph which is mapped by a morphism belonging to an edge in the corresponding class.

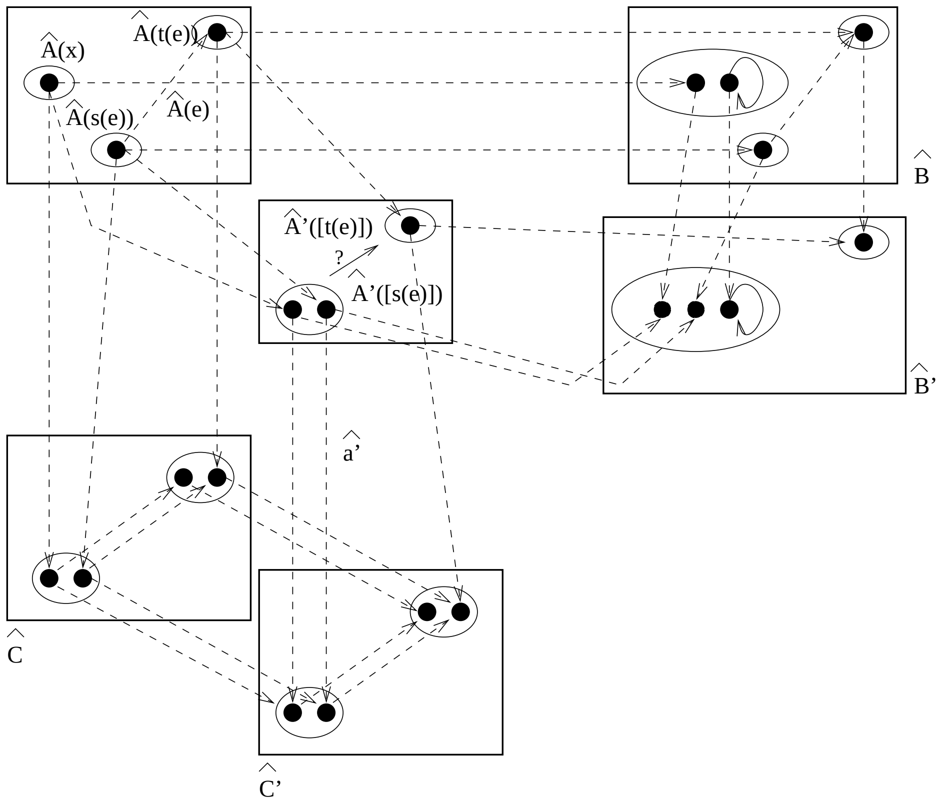

Figure 10. A gluing situation not allowed

Proposition 3.16 (Existence of Pushouts – 2). Given two D-graph morphisms $\hat{a} : \hat{A} \to \hat{C}$ and $\hat{b} : \hat{A} \to \hat{B}$ which satisfy the pushout conditions (1), (2) and (3) the pushout of $\hat{a}$ and $\hat{b}$ in **DISTR**(**GRAPH**) exists and can be constructed componentwise.

Proof sketch: Let D with $c : C \to D$ and $d : B \to D$ be the pushout of a and b. The pushout graph $\hat{D} : D \to \mathbf{GRAPH}$ is constructed in the following way (compare figure 3):

Let $\sim_{ab}$ be the equivalence relation induced by $\sim_a \cup \sim_b$.

1. For all $[x] \in A^N_{/\sim_{ab}}$ let $\hat{A}'([x])$ be the colimit graph of distributed graph $\hat{A}_{[x]}$. $\hat{f}_y : \hat{A}(y) \to \hat{A}'([x])$ for all $y \in [x]$ are the corresponding colimit morphisms.
2. For all $b([x]) \in B^N_{/b(\sim_{ab})}$ let $\hat{B}'(b([x]))$ be the colimit graph of distributed graph $\hat{B}_{b([x])}$. $\hat{g}_{b(y)} : \hat{B}(b(y)) \to \hat{B}'(b([x]))$ for all $b(y) \in b([x])$ are the corresponding colimit morphisms.
3. For all $a([x]) \in C^N_{/a(\sim_{ab})}$ let $\hat{C}'(a([x]))$ be the colimit graph of distributed graph $\hat{C}_{a([x])}$. $\hat{h}_{a(y)} : \hat{C}(a(y)) \to \hat{C}'(a([x]))$ for all $a(y) \in a([x])$ are the corresponding colimit morphisms.
4. Morphism $\hat{a}'_{[x]} : \hat{A}'([x]) \to \hat{C}'(a([x]))$ ($\hat{b}'_{[x]} : \hat{A}'([x]) \to \hat{B}'(b([x]))$) is the induced morphism of all $\hat{h}_{a(y)} \circ \hat{a}_y$ ($\hat{g}_{b(y)} \circ \hat{b}_y$), $\forall y \in [x]$, such that for all $x \in A^N$: $\hat{a}'_{[x]} \circ \hat{f}_y = \hat{h}_{a(y)} \circ \hat{a}_y$ ($\hat{b}'_{[x]} \circ \hat{f}_y = \hat{g}_{b(y)} \circ \hat{b}_y$), $\forall y \in [x]$.

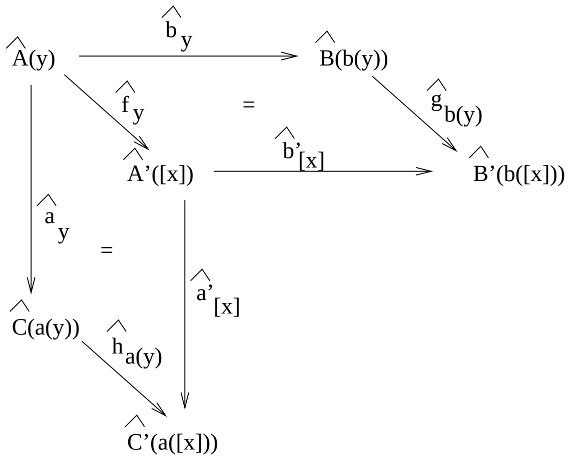

Figure 11. First steps of pushout construction

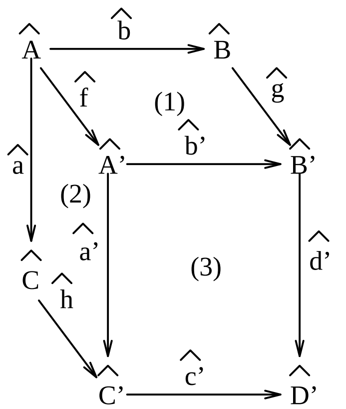

Figure 12. Construction of pushouts

5. For all $[e] \in A^E_{/\sim_{ab}}$ let $\hat{A}'([e]) : \hat{A}'([s(e)]) \to \hat{A}'([t(e)])$ be the induced morphism of $\hat{f}_{t(e')} \circ \hat{A}(e')$ for all $e' \in [e]$ such that $\hat{A}'([e]) \circ \hat{f}_{s(e')} = \hat{f}_{t(e')} \circ \hat{A}(e')$.

6. For all $[e] \in B^E_{/b(\sim_{ab})}$ let $\hat{B}'([e]) : \hat{B}'([s(e)]) \to \hat{B}'([t(e)])$ be the induced morphism of $\hat{g}_{t(e')} \circ \hat{B}(e')$ for all $e' \in [e]$ such that $\hat{B}'([e]) \circ \hat{g}_{s(e')} = \hat{g}_{t(e')} \circ \hat{B}(e')$.

7. For all $[e] \in C^E_{/a(\sim_{ab})}$ let $\hat{C}'([e]) : \hat{C}'([s(e)]) \to \hat{C}'([t(e)])$ be the induced morphism of $\hat{h}_{t(e')} \circ \hat{C}(e')$ for all $e' \in [e]$ such that $\hat{C}'([e]) \circ \hat{h}_{s(e')} = \hat{h}_{t(e')} \circ \hat{C}(e')$.

8. Construct the pushout over n-injective D-graph morphisms $\hat{a}' : \hat{A}' \to \hat{C}'$ and $\hat{b}' : \hat{A}' \to \hat{B}'$ according to proposition 3.10. This yields $(\hat{D}', \hat{c}' : \hat{C}' \to \hat{D}', \hat{d}' : \hat{B}' \to \hat{D}')$.

Let $\hat{c} = \hat{c}' \circ \hat{h}$ and $\hat{d} = \hat{d}' \circ g$. For illustrating the construction of the above graphs and graph morphisms see figure 3.

For the proof that $(\hat{D}', \hat{c}, \hat{d})$ is a pushout we first show that diagram (1) in figure 3 is a pushout:

- It is easy to show that (B', b', g) is the pushout of f and b. Since $\sim_b \subseteq \sim_f$ b' is injective.
- $\hat{A}'$ $(\hat{B}')$ is a D-graph since all $\hat{A}'(x)$ $(\hat{B}'(x))$ for $x \in A'^N$ $(x \in \hat{B}'^N)$ are colimit graphs, all $\hat{A}'(e)$ $(\hat{B}'(e))$ for $e \in A'^E$ $(x \in \hat{B}'^E)$ are induced graph morphisms and total according to pushout condition (3).
- $\hat{f} : \hat{A} \to \hat{A}'$ is a D-graph morphism since $\forall x \in A^N$, $\hat{f}_x$ is totally defined and $\forall e \in A^E$ we have $\hat{A}'(f(e)) \circ \hat{f}_{s(e')} = \hat{A}'([e]) \circ \hat{f}_{s(e')} = \hat{f}_{t(e')} \circ \hat{A}(e')$ according to the construction of $\hat{A}'([e])$.

- $\hat{g} : \hat{B} \rightarrow \hat{B}'$ is a D-graph morphism since $\forall x \in B^N$, $\hat{g}_x$ is totally defined and $\forall e \in B^E$ we have $\hat{B}'(g(e)) \circ \hat{g}_{s(e')} = \hat{B}'([e]) \circ \hat{g}_{s(e')} = \hat{g}_{t(e')} \circ \hat{B}(e')$ according to the construction of $\hat{B}'([e])$.

- According to pushout condition (3) and colimit properties it can be shown that $\hat{b}'$ is a D-graph morphism.

- We have $\forall x \in A^N : \hat{b}'_{f(x)} \circ \hat{f}_x = \hat{b}'_{[x]} \circ \hat{f}_x = \hat{g}_{b(x)} \circ \hat{b}_x$ since $\hat{b}'_{[x]}$ are induced morphisms of all $\hat{g}_{b(x)} \circ \hat{b}_x$. Moreover, we know $b' \circ f = g \circ b$, thus, $\hat{b}' \circ \hat{f} = \hat{g} \circ \hat{b}(*)$ follows.

- Given a D-graph $\hat{X}$ as well as two D-graph morphisms $\hat{k} : \hat{A}' \rightarrow \hat{X}$ and $\hat{l} : \hat{B} \rightarrow \hat{X}$ such that $\hat{k} \circ \hat{f} = \hat{l} \circ \hat{b}$ we have to show that there is a unique D-graph morphism $\hat{u} : \hat{B}' \rightarrow \hat{X}$ with $\hat{u} \circ \hat{b}' = \hat{k}$ and $\hat{u} \circ \hat{g} = \hat{l}$.

  u is the induced morphism of k and l such that $u \circ b' = k$ and $u \circ g = l$. $\forall [x] \in B'^N : \hat{u}_{[x]}$ is the induced morphism of all $\hat{l}_y$ such that $\hat{u}_{[x]} \circ \hat{g}_y = \hat{l}_y$ for all $y \in [x]$. Thus, $\hat{u}$ is totally defined. For $\hat{u}$ being a D-graph morphism $\forall [e] \in B'^E : \hat{u}_{[t(e)]} \circ \hat{B}'([e]) = \hat{X}(u([e])) \circ \hat{u}_{[s(e)]}$ can be shown using the colimit properties of $\hat{B}'([s(e)])$ $\forall e \in B^E$.

  As already mentioned above $\hat{u} \circ \hat{g} = \hat{l}(**)$ holds. To prove $\hat{u} \circ \hat{b}' = \hat{k}$ we have to show that $\hat{u} \circ \hat{b}'$ as well as $\hat{k}$ are induced morphisms of all $l_{b(x)} \circ b_x$ for $x \in A^N$ such that $\hat{k} \circ \hat{f} = \hat{l} \circ \hat{b} = \hat{u} \circ \hat{g} \circ \hat{b} = \hat{u} \circ \hat{b}' \circ \hat{f}$ due to equations (*) and (**). This is the case if $\forall e \in A^E_{[s(e)]} : \hat{l}_{b(s(e))} \circ \hat{b}_{s(e)} = \hat{l}_{b(t(e))} \circ \hat{b}_{t(e)} \circ \hat{A}(e)$. But this is true since $\hat{b}$ and $\hat{l}$ are D-graph morphisms.

  Uniqueness of $\hat{u}$ follows directly from the definition of $\hat{u}$ where u and $\hat{u}_x$ $\forall x \in B'^N$ are chosen as induced ones.

- Analogously, it can be shown that $(\hat{C}', \hat{a}', \hat{h})$ is the pushout of $\hat{a}$ and f. (Compare pushout (2) in figure 3.)

- Pushout conditions (1) and (2) which are satisfied for $\hat{a}$ and $\hat{b}$ by assumption are also satisfied by $\hat{a}'$ and $\hat{b}'$ since context edges $e \in B^E - b(A^E)$ and $e \in C^E - a(A^E)$ are isomorphically mapped into $\hat{B}'$ and $\hat{C}'$. Thus, $(\hat{D}', \hat{c}', \hat{d}')$ is a pushout of $\hat{a}'$ and $\hat{b}'$ due to proposition 3.10.

- Lastly, we have to show that $(\hat{D}', \hat{c}, \hat{d})$ with $\hat{c} = \hat{c}' \circ \hat{h}$ and $\hat{d} = \hat{d}' \circ \hat{g}$ is the pushout of $\hat{a}$ and $\hat{b}$. This can be easily shown based on colimit properties of pushouts (1),(2) and (3) in figure 3 and colimits $(\hat{A}'([x]), (\hat{f}_y)_{y \in [x]})$, $\forall [x] \in A'^N$.

□

Example 3.17 (Pushout in **DISTR(GRAPH)***- 2).* In figure 3.17 an example for a pushout over distributed graph morphisms is shown where one morphism is not n-injective. Pushout conditions (1) and (2) are trivially satisfied since there are no context edges in network graphs C and B. Pushout condition (3) is satisfied since gluing $\hat{A}(x)$ and $\hat{A}([s(e)])$ via $\hat{A}(y)$ yields the same result as a gluing without $\hat{A}(y)$. The same holds for $\hat{B}$. In $\hat{C}$ nothing is glued. Thus, $\hat{C}'$ is isomorphic to $\hat{C}$. $\hat{A}'([e])$ is the induced morphism wrt. $\hat{A}'([s(e)])$ where only $\hat{A}(e)$ has to be taken into account. Similarly $\hat{B}'$ is constructed. Morphism $\hat{a}'_{[s(e)]}$ is the induced morphism out of $\hat{A}'([s(e)])$ taking $\hat{a}_{s(e)}$, $\hat{a}_x$ and $\hat{a}_y$ into account. $\hat{a}'_{[t(e)]}$ is just the copy of $\hat{a}_{t(e)}$. Similarly $\hat{b}'$ is constructed. Since $\hat{a}'$ and $\hat{b}'$ are n-injective pushout graph $\hat{D}$ with morphisms $\hat{c}'$ and $\hat{d}'$ can be constructed as explained above in the chapter.

Proposition 3.18 (Componentwise PO-construction in **DISTR(GRAPH)***-2).* Given two D-graph morphisms $\hat{a} : \hat{A} \rightarrow \hat{C}$ and $\hat{b} : \hat{A} \rightarrow \hat{B}$, the pushout of $\hat{a}$ and $\hat{b}$ in **DISTR(GRAPH)** can be constructed componentwise if and only if pushout conditions (1), (2) and (3) are satisfied.

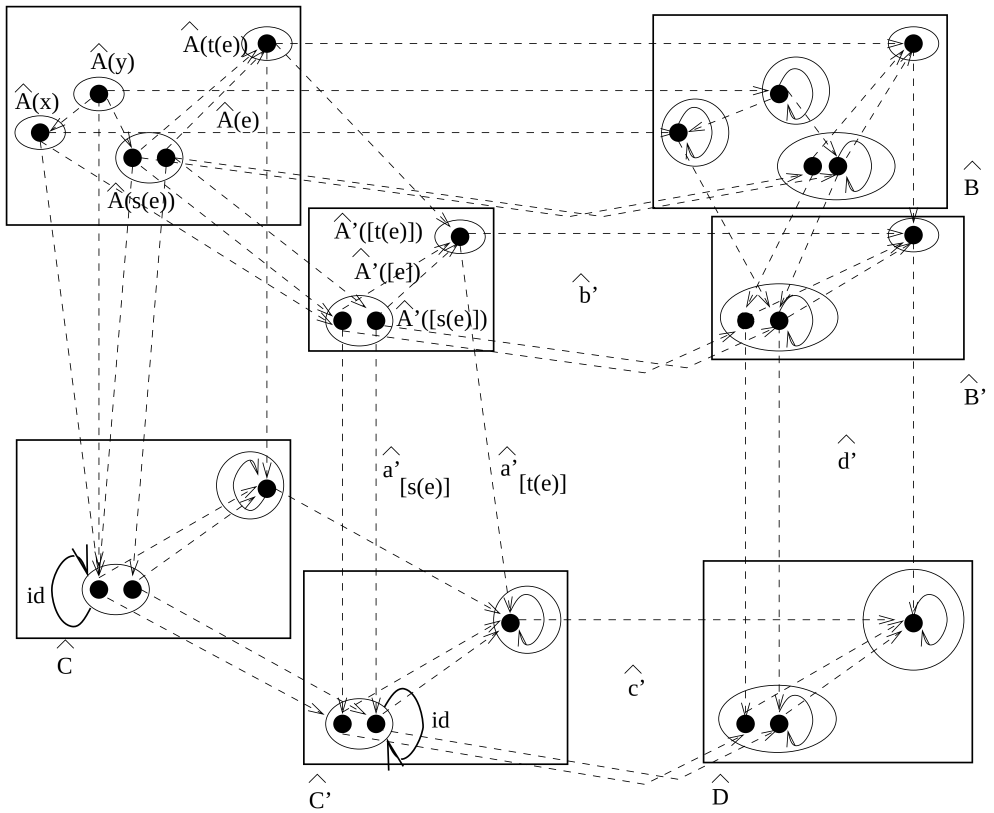

Figure 13. Example of a pushout in **DISTR(GRAPH)**– 2

Proof: Proposition 3.12 states that pushout conditions (1) and (2) are necessary and sufficient for the componentwise construction of the pushout over n-injective D-graph morphisms. Proposition 3.16 shows that pushout conditions (1), (2) and (3) are sufficient for the componentwise construction of pushouts over arbitrary D-graph morphisms. It remains to show that pushout condition (3) is also necessary for the componentwise pushout construction. Assuming that pushout condition (3) is not satisfied, $\hat{A}$ with $\equiv (\sim_a \cup \sim_b)$ does not satisfy the network gluing condition, for example. This means that there is an $x \in A^N$ with $x \in [s(e)]$ for some $e \in A^E$ and $\not\exists e' \in [e]$ with $s(e') = x$ such that there is a $[y] \in \hat{A}'([s(e)])$ where a $y' \in [y]$ with $y' \in \hat{A}(s(e))$ does not exist. (consider figure 3.15 for such a situation.) Since $[y]$ contains at least y it is not empty. Since $\hat{A}'([e])$ is the induced morphism of $\hat{f}_{t(e')} \circ \hat{A}(e')$ for all $e' \in [e]$, y is not mapped by $\hat{A}'([e])$. Hence, $\hat{A}'([e])$ is not total. The proofs for the necessity of pushout conditions (3)(b) and (3)(c) can be formulated analogously to the proof above. □

Considering pushout situations in figures 9 and 10 , distributed graphs $\hat{D}$ can become pushout graphs if local graphs $\hat{C}(t(e))$ are not identically mapped to $\hat{D}(c(t(e)))$ but $\hat{c}_{t(e)}$ has to reflect $\hat{c}_{s(e)}$. Moreover, a class $[y]$ in $\hat{A}'([s(e)])$ where $\not\exists y' \in [y]$ with $y' \in \hat{A}(s(e))$ has to have an image in $\hat{A}'([t(e)])$. This observation leads to the conjecture that pushouts exist for each two D-graph morphisms $\hat{a} : \hat{A} \to \hat{C}$ and $\hat{b} : \hat{A} \to \hat{B}$ in category **DISTR(GRAPH)**, but if they do not satisfy the pushout conditions they cannot be constructed componentwise, i.e. strictly modular. Since the componentwise construction reflects the distributed computation best, pushouts for D-graph morphisms $\hat{a}$ and $\hat{b}$ in general are not investigated in this paper.

4. Distributed Graph Transformation

Distributed graph transformation is described in the framework of the double-pushout approach to graph transformation which has been reviewed in section 2. This approach seems to be appropriate to be extended by distribution concepts since it provides some kind of cautious graph rewriting. This means that the structure of distributed graphs is not destroyed during transformation and side-effects are not performed.

Distributed graph transformation where a distributed structure of graphs is transformed by local graph transformations can be characterized by a double-pushout over distributed graphs and graph morphisms, i.e. in the category **DISTR(GRAPH)**. The double-pushout approach has been generalized to so-called high-level replacement systems in (EHKP91; EL93) where the main results of the graph grammar theory are abstracted to arbitrary objects and morphisms. In the following we show that distributed graph transformation fits into the framework of HLR-systems.

Definition 4.1 (D-graph production). A distributed graph production, short *D-graph production*, $\hat{p} = (\hat{L} \xleftarrow{\hat{l}} \hat{I} \xrightarrow{\hat{r}} \hat{R})$ consists of D-graphs $\hat{L} : L \to \mathbf{GRAPH}$, $\hat{R} : R \to \mathbf{GRAPH}$ and $\hat{I} : I \to \mathbf{GRAPH}$, called left- and right-hand side and intermediate D-graphs and two D-graph morphisms $\hat{l}$ and $\hat{r}$.

- If all $\hat{p}_x = (\hat{L}(l(x)) \xleftarrow{\hat{l}_x} \hat{I}(x) \xrightarrow{\hat{r}_x} \hat{R}(r(x)))$ for $x \in I^N$ are left-injective, i.e. all $\hat{l}_x$ are injective, $\hat{p}$ is called *left-injective*, too.
- If $L \cong I \cong R$ are network graphs consisting of exactly one node each, $\hat{p}$ is called *local*.
- If $L \cong I \cong R$, $\hat{p}$ is called *synchronized*.
- If $\hat{p}_x$ is identical (def. 2.4), $\forall x \in I_N$, $\hat{p}$ is called *network transforming production*.

A local production describes a local action, communication and synchronization can be modeled by synchronized productions and network administration is expressed by network transforming productions.

Definition 4.2 ((Extended) distributed gluing condition and D-match). Given a D-graph production $\hat{p} = (\hat{L} \xleftarrow{\hat{l}} \hat{I} \xrightarrow{\hat{r}} \hat{R})$ a D-graph morphism $\hat{m} : \hat{L} \to \hat{G}$, called *D-graph occurrence*, satisfies the following *distributed gluing condition* if:

1. *global gluing condition*: $m : L \to G$ satisfies the *gluing condition* wrt. $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ (compare the gluing condition in definition 2.5).
2. *local gluing conditions*: $\forall x \in I^N : \hat{m}_{l(x)}$ satisfies the *gluing condition* wrt. $\hat{p}_x = (\hat{L}(l(x)) \xleftarrow{\hat{l}_x} \hat{I}(x) \xrightarrow{\hat{r}_x} \hat{R}(r(x)))$ and all $\hat{p}_x$ are l-injective.
3. *connection condition*: Let $Context(q, n) = H - n(L') + n(l'(I'))$ be the context graph of production $q = (L' \xleftarrow{l'} I' \xrightarrow{r'} R')$ and match $n : L' \to H$:
   (a) $\forall x, y \in I^N \forall e : m(l(x)) \to m(l(y)) \in Context(p, m)^E$:[4]

$$\hat{G}(e)(Context(\hat{p}_x, \hat{m}_{l(x)})) \subseteq Context(\hat{p}_y, \hat{m}_{l(y)})$$

   $\forall y \in I^N \forall e : z \to m(l(y)) \in G^E : \hat{G}(e)(\hat{G}(z)) \subseteq Context(\hat{p}_y, \hat{m}_{l(y)})$
   (b) $\forall x \in I^N \forall e : m(l(x)) \to z \in G^E - m(L^E) : \hat{l}_y$ and $\hat{r}_y$ are bijective.

[4] We write $e : a \to b$ for $s(e) = a$ and $t(e) = b$.

4. *network condition*: (a) $\forall x \in L^N - l(I^N) : \hat{m}_x$ is bijective
 $\forall x \in I^N \forall e : l(x) \rightarrow y \in L^E - l(I^E) : \hat{m}_x$ is bijective
 (b) $\forall x \in I^N \forall e : r(x) \rightarrow y \in R^E - r(I^E) : \hat{m}_{l(y)}$ is bijective.

$\hat{m}$ is said *to satisfy all local identification (dangling) conditions* if $\hat{m}_{l(x)}$ satisfies the identification (dangling) condition wrt. $\hat{p}_x$, $\forall x \in \hat{I}^N$.

D-graph morphism $\hat{m} : \hat{L} \rightarrow \hat{G}$ is a *D-match of* $\hat{p}$ if the following *extended distributed gluing condition* is satisfied in additional:

Let $\hat{I}'$ be a subgraph of $\hat{I}$ such that $I' = SPLIT(p)$ and $\sim_{m'} = \{(x, y) \in I \times I | m(l(x)) = m(l(y)) \wedge (x \neq y \Longrightarrow x, y \notin SPLIT(p))\}$.

1. *global extended gluing condition*: $m : L \rightarrow G$ satisfies the *extended gluing condition* wrt. $p = (L \xleftarrow{l} I \xrightarrow{r} R)$ (compare the gluing condition in definition 2.5).
2. *splitting condition*: (a) $\forall x \in l(SPLIT(p)) : \hat{m}_x$ is bijective or (b) $\hat{I}'$ with $equiv(\sim_l)$ satisfies the network gluing condition and $forall y \in SPLIT(p)$ with $l(y) = x$ and $[y] \in I^N_{/\sim_l}$: The network graph of $\hat{I}_{[y]}$ has to be connected.
3. *local parallelism condition:* $\forall x \in L^N$ with $m(x) = m(y)$ for some $y \in L^N : \hat{m}_x$ is bijective or (a) $\hat{I}$ with $equiv(\sim_{m'})$ satisfies the network gluing condition. (b) $\hat{L}$ with $equiv(\sim_m)$ satisfies the network gluing condition. (c) $\hat{R}$ with $equiv(r(\sim_{m'}))$ satisfies the network gluing condition.
4. *join condition*: (a) $\hat{I}$ with $equiv(\sim_{m'} \cup \sim_r)$ satisfies the network gluing condition. (b) $\hat{G}$ with $equiv(m(l(\sim_r)))$ satisfies the network gluing condition.

The *connection condition* means the following: (a) An action on a target graph is not permitted to delete local items if some of their counterparts in connected source graphs are not deleted. (b) A local action on a source graph is not allowed to extend this graph if new items would not have images in connected target graphs. Furthermore, such an action is not allowed to delete local items without deleting references to items in target graphs. Lastly, local objects in a source graph are not allowed to be glued together if this is not reflected in the connected target graphs.

The *network condition* can be interpreted as follows: (a) The deletion of network nodes can be done if its local state graph is deleted as a whole in the same production, i.e. if the current local state corresponds with that in the production. If a network edge should be deleted, its source graph has to correspond bijectively with that of the production. (b) The other way around, if a new connection from an existing source graph should be established, this graph has to be structural equivalent with its correspondent given in the production.

The *splitting condition* describes permitted splitting situations: Graphs which should be split have to be fully determined by the match or they are split in a way such that the local parts that are copied to the split graphs are related by interface graphs. For a splitting situation not allowed consider the next example 4.3.

The *local parallelism condition* expresses when local productions can be applied in parallel: Productions which should be applied in parallel to a local graph have to cover this graph completely or different parallel production applications on interface graphs have to harmonize with each others.

The *join condition* states when local graphs may be joined.

Example 4.3 (A Splitting situation not allowed). Considering figure 4.3 a situation is shown where the local graph of $\hat{L}$ with one node should be split into two independent local graphs in $\hat{I}$ such that the node is copied. Using a match $\hat{m}$ that embeds the node into larger context this context is copied into both local graphs of D-graph $\hat{C}$. (See the following definition 4.4 of context graphs.) Following the pushout construction for $\hat{l}$ and $\hat{c}$ as described in proposition 3.16 we get D-graph $\hat{G}'$ and not G.

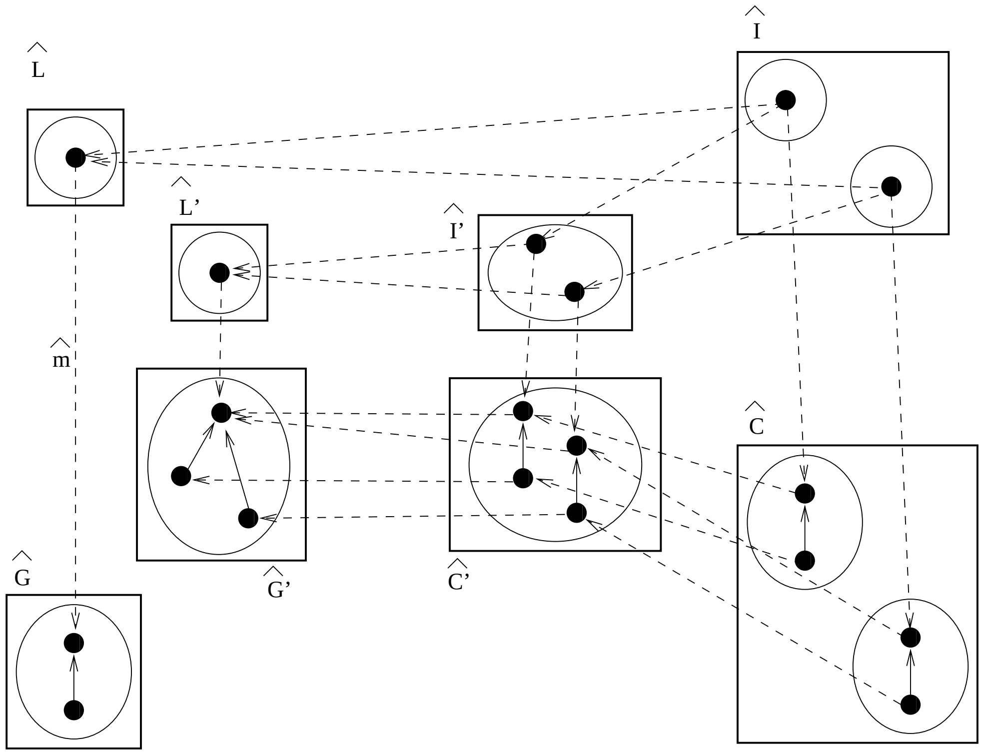

Figure 14. A splitting situation not allowed

The problem depicted here is a general splitting problem: If the graph is not completely split, the remaining context can be split, too, in an arbitrary way leading to some kind of non-determinism. Otherwise the context has to be copied into all parts involved. But then, the information about this copy operation is lost if there is not an interface graph for the split part that keeps this information.

Definition 4.4 (D-context graph). Given a D-graph production $\hat{p} = (\hat{L} \xleftarrow{\hat{l}} \hat{I} \xrightarrow{\hat{r}} \hat{R})$ and a D-match $\hat{m} : \hat{L} \to \hat{G}$, the following D-graph $\hat{C} : C \to \mathbf{GRAPH}$ is called *D-context graph* of $\hat{p}$ and $\hat{m}$. Let $C = G - m(L - l(I)) - m(l(SPLIT(p))) + SPLIT(p)$ be the context graph of p and m in **GRAPH**.

$$\hat{C}(x) := \begin{cases} \text{the largest subgraph of} & \\ \hat{G}(g(x)) - \bigcup_{y \in \{z | c(z) = x\}} \hat{m}_{l(y)}(\hat{L}(l(y)) - \hat{l}_y(\hat{I}(y))) & , \text{ if } x \in C^N \\ \hat{g}^{-1}_{t(x)} \circ \hat{G}(g(x)) \circ \hat{g}_{s(x)} & , \text{ if } x \in C^E \end{cases}$$

where D-graph morphisms $\hat{g} : \hat{C} \to \hat{G}$ and $\hat{c} : \hat{I} \to \hat{C}$ are defined as follows. Let $c = m \circ l$ and $g = id_{G|_C}$, D-graph morphism $\hat{c}$ is defined by $\hat{c}_y = \hat{m}_{l(y)} \circ \hat{l}_y$, $\forall y \in I^N$, and D-graph morphism $\hat{g}_x = id_{\hat{G}(x)_{/\hat{C}(x)}}$, $\forall x \in G^N$[5] and $\hat{g}_x = \hat{m}_{l(x)} \circ \hat{l}_x$ otherwise.

Remark: $\forall y \in I^N$ the PO-complement $(\hat{C}(c(y)), \hat{g}_{c(y)}, \hat{c}_y)$ of $\hat{m}_{l(y)}$ and $\hat{l}_y$ in **GRAPH** is constructed.

Proposition 4.5 (Applicability of D-graph productions - 1). Given a left-injective D-graph production $\hat{p} = (\hat{L} \xleftarrow{\hat{l}} \hat{I} \xrightarrow{\hat{r}} \hat{R})$, a D-match $\hat{m} : \hat{L} \to \hat{G}$ and the D-context graph $\hat{C}$ as well as D-graph morphisms $\hat{g}$ and $\hat{c}$ as defined in 4.4, then $(\hat{G}, \hat{m}, \hat{g})$ is the pushout of $\hat{c}$ and $\hat{l}$ in **DISTR(GRAPH)**.

Proof sketch: In this special case, $C = G - m(L - l(I))$ and $\hat{C}(x) = \hat{G}(x) - \hat{m}_{l(y)}(\hat{L}(l(y)) - \hat{l}_y(\hat{I}(y)))$, $\forall y \in I^N$ with $m \circ l(y) = x$.

[5] where $id_{\hat{G}(x)_{/\hat{G}(x)}} = id_{\hat{G}(x)}$

$$\begin{tikzcd}
\hat{L} \arrow[d, "\hat{m}"] \arrow[dd, bend right, "\hat{f}"'] & \hat{I} \arrow[l, "\hat{l}"'] \arrow[d, "\hat{c}"] \\
\hat{G} \arrow[d, shift left, "\hat{w}"] & \hat{C} \arrow[l, "\hat{g}"'] \arrow[dl, "\hat{h}"] \\
\hat{X} \arrow[u, shift left, "\hat{u}"] &
\end{tikzcd}$$

Figure 15. Illustration for the proof of proposition 4.5

$\hat{C}$ is a D-graph since the gluing condition for all local transformations and the connection condition (a) are satisfied. This means for each $x \in C^N$, $\hat{C}(x)$ is a graph, since the gluing condition is satisfied for the application of each local production $\hat{p}_x$ at match $\hat{m}_{l(x)}$. For each $e \in C^E$, $\hat{C}(e)$ is a total graph morphism, since the connection condition (a) is satisfied for e wrt. $\hat{p}_{s(e)}$, $\hat{p}_{t(e)}$, $\hat{m}_{s(e)}$ and $\hat{m}_{t(e)}$. Thus, $\forall x \in \hat{C}(s(e))$, there is a $y \in \hat{C}(t(e))$ with $\hat{G}(e) \circ \hat{g}_{s(e)}(x) = y$. It is easy to show that $\hat{g}$ is a D-graph morphisms.

Now we have to show the pushout properties for $(\hat{G}, \hat{m}, \hat{g})$. The commutativity $\hat{m} \circ \hat{l} = \hat{g} \circ \hat{c}$ follows from fact 2.7. Next, we construct the pushout of $\hat{l}$ and $\hat{c}$ (which is possible since the distributed gluing conditions contain the pushout conditions which are part of network condition (a) and connection condition (b)) and obtain $(\hat{X}, \hat{f} : \hat{L} \to \hat{X}, \hat{h} : \hat{C} \to \hat{X})$. Then, we have to show that the resulting pushout graph $\hat{X}$ is isomorphic to $\hat{G}$. According to pushout properties there is a unique D-graph morphism $\hat{u} : \hat{X} \to \hat{G}$ with $\hat{u} \circ \hat{f} = \hat{m}$ and $\hat{u} \circ \hat{h} = \hat{g}$. Vice versa, a suitable D-graph morphism $\hat{w} : \hat{G} \to \hat{X}$ can be defined as follows. $\forall x \in m(l(I^N))$, $\hat{w}_x$ is the induced morphism wrt. $(\hat{G}(x), \hat{m}_x, \hat{g}_x)$. $\forall z \in C^N - c(I^N)$ with $g(z) = x$, $\hat{w}_x = \hat{h}_z$ and $\forall v \in L^N - l(I^N)$ with $m(v) = x$, $\hat{w}_x = \hat{f}_v \circ m_v^{-1}$. (Compare the diagram in figure 4.) The well-definedness of $\hat{w}$ can be shown based on network condition (a).

It is easy to show that $\hat{w} \circ \hat{g} = \hat{h}$ and $\hat{w} \circ \hat{m} = \hat{f}$ hold. $\hat{w}_z \circ \hat{g}_z = \hat{h}_z$, $\forall z \in C^N$ and $\hat{w}_v \circ \hat{m}_v = \hat{f}_v$, $\forall v \in L^N$, follow directly from PO-properties or the definition of $\hat{w}$. It is straight forward to show $\hat{u} \circ \hat{w} = id_{\hat{G}}$ and $\hat{w} \circ \hat{u} = id_{\hat{X}}$. Thus, $\hat{G}$ and $\hat{X}$ are isomorphic.

□

Proposition 4.6 (Applicability of D-graph productions - 2). Given a left-surjective[6] D-graph production $\hat{p} = (\hat{L} \xleftarrow{\hat{l}} \hat{I} \xrightarrow{\hat{r}} \hat{R})$ and a D-match $\hat{m} : \hat{L} \to \hat{G}$ there are a D-context graph $\hat{C}$ as well as D-graph morphisms $\hat{g}$ and $\hat{c}$ as defined in 4.4 such that $(\hat{G}, \hat{g}, \hat{m})$ is the pushout of $\hat{l}$ and $\hat{c}$ in **DISTR(GRAPH)**.

Proof sketch: (C, g, c) exists and forms the initial PO-complement over l and m in **GRAPH** according to proposition 2.10.

Since $\hat{p}$ is left-surjective, the definition of the context graph $\hat{C}$ and of $\hat{g}$ can be simplified: $\hat{C}(x) = \hat{G}(g(x))$ and $\hat{g}_x = id_{\hat{G}(g(x))}$, $\forall x \in C^N$. The network edges of $\hat{C}$ and D-graph morphism $\hat{c}$ are defined as in 4.4. Obviously, $(\hat{C}(c(y)), \hat{g}_{c(y)}, \hat{c}_y)$ is an initial pushout complement $\forall y \in I^N$ (compare proposition 2.10).

Furthermore, it is obvious that $\hat{C}$ is a D-graph and $\hat{g}$ as well as $\hat{c}$ are D-graph morphisms. Next we show the pushout properties for $(\hat{G}, \hat{g}, \hat{m})$. The commutativity follows directly. Then, we construct the pushout of $\hat{l}$ and $\hat{c}$. This is possible since PO-conditions (1) and (2) are captured by the distributed gluing condition, namely by network condition (a) and connection condition (b). Pushout condition (3) is satisfied, too, because of the splitting and the local parallelism condition.

[6] This means that all $\hat{l}_x$ are bijective.

$$\begin{tikzcd} \hat{L} \arrow[d, "\hat{m}"'] \arrow[dr, phantom, "(1)"] & \hat{I} \arrow[l, "\hat{l}"'] \arrow[r, "\hat{r}"] \arrow[d, "\hat{c}"] \arrow[dr, phantom, "(2)"] & \hat{R} \arrow[d] \\ \hat{G} & \hat{C} \arrow[l, "\hat{g}"'] \arrow[r] & \hat{H} \end{tikzcd}$$

Figure 16. D-graph transformation

Let $(\hat{X}, \hat{f} : \hat{L} \to \hat{X}, \hat{h} : \hat{C} \to \hat{X})$ be the pushout of $\hat{l}$ and $\hat{c}$. Now, we have to show that pushout graph $\hat{X}$ is isomorphic to $\hat{G}$. According to pushout properties there is a unique D-graph morphism $\hat{u} : \hat{X} \to \hat{G}$ with $\hat{u} \circ \hat{f} = \hat{m}$ and $\hat{u} \circ \hat{h} = \hat{g}$. Vice versa, a suitable D-graph morphism $\hat{w} : \hat{G} \to \hat{X}$ can be defined as follows:

Since $\hat{l}_x$ is bijective, $\forall x \in I^N$, $\hat{C}(y) = \hat{G}(g(y))$, $\forall y \in C^N$. Thus, $(\hat{G}(m(l(x))), \hat{g}_{m(l(x))} = id_{\hat{G}(m(l(x)))}, \hat{m}_{l(x)})$ is the pushout of $\hat{l}_x$ and $\hat{c}_x$, $\forall x \in I^N$. Therefore, $\forall y = m(l(x))$, $x \in I^N$, morphism $\hat{w}_y$ can be defined as the induced morphism wrt. the pushout of $\hat{l}_x$ and $\hat{c}_x$. $\forall z \in C^N - c(I^N)$ with $g(z) = y$ we define $\hat{w}_y = \hat{h}_z$. The well-definedness of $\hat{w}$ follows from the universal property of pushouts.

It is easy to show that $\hat{w} \circ \hat{g} = \hat{h}$ and $\hat{w} \circ \hat{m} = \hat{f}$ hold. Moreover, it is straight forward to show $\hat{u} \circ \hat{w} = id_{\hat{G}}$ and $\hat{w} \circ \hat{u} = id_{\hat{X}}$. Thus, $\hat{G}$ and $\hat{X}$ are isomorphic. □

Definition 4.7 (Quasi-initial pushout complement). A pushout complement $(\hat{C}, \hat{c} : \hat{I} \to \hat{C}, \hat{g} : \hat{C} \to \hat{G})$ of $\hat{l} : \hat{I} \to \hat{L}$ and $\hat{m} : \hat{L} \to \hat{G}$ in **DISTR(GRAPH)**is called *quasi-initial* if

1. (C, c, g) is the initial pushout complement of l and m in **GRAPH**.
2. $\forall x \in I^N$: $(\hat{C}(c(x)), \hat{c}_x, \hat{g}_{c(x)})$ is the initial pushout complement over $\hat{m}_{l(x)}$ and $\hat{l}_x$ in **GRAPH**.
3. $\forall x \in C^N - c(I^N)$: $\hat{g}_x$ is the identity.

Proposition 4.8 (Uniqueness of PO-complement). The pushout complement $(\hat{C}, \hat{c} : \hat{I} \to \hat{C}, \hat{g} : \hat{C} \to \hat{G})$ of $\hat{l} : \hat{I} \to \hat{L}$ and $\hat{m} : \hat{L} \to \hat{G}$ as constructed in proposition 4.6 is unique up to isomorphism in the class of all quasi-initial pushout complements of $\hat{l}$ and $\hat{m}$.

Proof sketch: Comparing the proof of proposition 4.6 it is easy to show that $(\hat{C}, \hat{c} : \hat{I} \to \hat{C}, \hat{g} : \hat{C} \to \hat{G})$ is quasi-initial. Assume another quasi-initial pushout complement $(\hat{C}', \hat{c}' : \hat{I} \to \hat{C}', \hat{g}' : \hat{C}' \to \hat{G})$ of $\hat{l}$ and $\hat{m}$. There are graph morphisms $h : C \to C'$ and $h' : C' \to C$ with $h \circ h' = id_C$ and $h' \circ h = id_{C'}$ because of property 1 in definition 4.7. Thus, graph C is isomorphic to graph C'. There are graph morphisms $\hat{h}_x : \hat{C}(x) \to \hat{C}'(h(x))$ and $\hat{h}'_{h(x)} : \hat{C}'(h(x)) \to \hat{C}(x)$, $\forall x \in c(I^N)$. We have $\hat{h}_x \circ \hat{h}'_{h(x)} = id_{\hat{C}(x)}$ and $\hat{h}'_{h(x)} \circ \hat{h}_x = id_{\hat{C}'(h(x))}$ for all pairs $\hat{h}_x$ and $\hat{h}'_{h(x)}$ because of property 2 of definition 4.7. Property 3 in definition 4.7 and the properties of initial pushout complements are used to show that $\hat{h}$ and $\hat{h}'$ are D-graph morphisms. $\hat{h} \circ \hat{h}' = id_{\hat{C}}$ and $\hat{h}' \circ \hat{h} = id_{\hat{C}'}$ follows from the statements above. Thus, $\hat{C}$ and $\hat{C}'$ are isomorphic to each other. □

Definition 4.9 (D-graph transformation (sequence)). Given a D-graph production $\hat{p} = (\hat{L} \xleftarrow{\hat{l}} \hat{I} \xrightarrow{\hat{r}} \hat{R})$ and a D-match $\hat{m} : \hat{L} \to \hat{G}$, a distributed graph transformation, short *D-graph transformation*, $\hat{G} \Longrightarrow_{di} \hat{H}$ via $\hat{p}$ and $\hat{m}$, short $\hat{G} \overset{\hat{p},\hat{m}}{\Longrightarrow}_{di} \hat{H}$, from a D-graph $\hat{G}$ to a D-graph $\hat{H}$ is given by the two pushout diagrams (1) and (2) in the category **DISTR**(**GRAPH**) shown in figure 4.9.

A *D-graph transformation sequence* $\hat{G} \overset{P}{\Longrightarrow}{}^*_{di} \hat{H}$ is a sequence of $n \geq 0$ D-graph transformations $\hat{G} = \hat{G}_0 \Longrightarrow_{di} \hat{G}_1 \Longrightarrow_{di} \ldots \Longrightarrow_{di} \hat{G}_n = \hat{H}$ via D-graph productions of a set P. $\hat{H}$ is also called *D-derivable* from $\hat{G}$ by P.

Proposition 4.10 (Applicability and uniqueness of D-graph transformation). Given a D-graph production $\hat{p} = (\hat{L} \xleftarrow{\hat{l}} \hat{I} \xrightarrow{\hat{r}} \hat{R})$ and a D-match $\hat{m} : \hat{L} \to \hat{G}$, the D-graph transformation $\hat{G} \stackrel{\hat{p},\hat{m}}{\Longrightarrow}_{di} \hat{H}$ exists and is unique up to isomorphism.

Proof sketch: According to proposition 3.7 D-graph morphism $\hat{l}$ can be uniquely decomposed into an epimorphism $\hat{l}_e : \hat{I} \to \hat{I}_{\equiv}$ and a monomorphism $\hat{l}_i : \hat{I}_{\equiv} \to \hat{L}$. Since all $\hat{l}_x$ for $x \in I^N$ are injective, all $\hat{l}_{ex}$ are bijective. Looking at D-graph morphisms $\hat{m}$ and $\hat{l}_i$ there is a PO-complement $(\hat{C}', \hat{g}_i : \hat{C}' \to \hat{G}, \hat{c}' : \hat{I}_{\equiv} \to \hat{C}')$ using proposition 4.5. Since $\hat{m}$ satisfies the extended distributed gluing condition wrt. $\hat{p}$ it can be easily shown that $\hat{c}'$ also satisfies it wrt. $\hat{p}' = (\hat{I}_{\equiv} \xleftarrow{\hat{l}_e} \hat{I} \xrightarrow{\hat{r}} \hat{R})$. Thus, there is a quasi-initial PO-complement $(\hat{C}, \hat{c} : \hat{I} \to \hat{C}, \hat{g}_e : \hat{C} \to \hat{C}')$ according to proposition 4.6 and 4.8. Hence, the pushout $(\hat{G}, \hat{m}, \hat{g} = \hat{g}_i \circ \hat{g}_e)$ of $\hat{l}$ and $\hat{c}$ exists and is uniquely determined.

The pushout of $\hat{c}$ and $\hat{r}$ exists according to proposition 3.16. Pushout conditions (1) is part of connection condition (b), pushout condition (2) corresponds to network (b) and pushout condition (3) corresponds to the local parallelism condition (c) and the join condition. Together with the fact that pushouts in arbitrary categories are unique up to isomorphisms we can state that $G \stackrel{\hat{p},\hat{m}}{\Longrightarrow}_{di} H$ is unique up to isomorphism. □

Distributed actions are supposed to run simultaneously. Such a parallel execution of distributed actions can be expressed by a parallel production on distributed graphs.

Definition 4.11 (Parallel D-graph production). Given two D-graph productions $\hat{p}_1 = (\hat{L}_1 \xleftarrow{\hat{l}_1} \hat{I}_1 \xrightarrow{\hat{r}_1} \hat{R}_1)$ and $\hat{p}_2 = (\hat{L}_2 \xleftarrow{\hat{l}_2} \hat{I}_2 \xrightarrow{\hat{r}_2} \hat{R}_2)$, the D-graph production $\hat{p}_1 + \hat{p}_2 = (\hat{L}_1 + \hat{L}_2 \xleftarrow{\hat{l}_1+\hat{l}_2} \hat{I}_1 + \hat{I}_2 \xrightarrow{\hat{r}_1+\hat{r}_2} \hat{R}_1 + \hat{R}_2)$ is called *parallel D-graph production* of $\hat{p}_1$ and $\hat{p}_2$ where $\hat{L}_1 + \hat{L}_2$, $\hat{I}_1 + \hat{I}_2$ and $\hat{R}_1 + \hat{R}_2$ are coproduct graphs and $\hat{l}_1 + \hat{l}_2$ as well as $\hat{r}_1 + \hat{r}_2$ the induced morphisms.

The construction of parallel productions can be iterated[7] leading to productions $\hat{p}_1 + \hat{p}_2 + \ldots + \hat{p}_n$ with $n \geq 1$ which are also called parallel productions, written $\sum_1^n \hat{p}_i = (\sum_1^n \hat{L}_i \xleftarrow{\sum \hat{l}_i} \sum_1^n \hat{I}_i \xrightarrow{\sum \hat{r}_i} \sum_1^n \hat{R}_i)$.

Definition 4.12 (Distributed graph grammar and its semantics). A *distributed graph grammar* $DGG = (\hat{S}, P)$ is given by a D-graph $\hat{S}$, the *start graph* and a set P of D-graph productions.[8]

Let P^+ be the smallest extension of the set P including all parallel D-graph productions $\hat{p}_1 + \hat{p}_2$ for $\hat{p}_1, \hat{p}_2 \in P^+$. The *operational semantics* $Sem(DGG)$ of DGG is given by the class of all possible transformations starting at $\hat{S}$ using D-graph production set P^+, i.e. $Sem(DGG) = \{\hat{S} \stackrel{P^+}{\Longrightarrow}{}^*_{di} \hat{G}\}$.

As operational semantics we take all possible distributed graph transformations where distributed productions may be applied in parallel.

5. Conclusion

Modeling distributed systems by distributed graph transformation a powerful and flexible description of *dynamic networks* is obtained by applying graph transformation to network structures. *Communication* and *synchronization* are modeled by graph sharing and common subtransformation on shared graph parts.

[7] Please note that the construction of a parallel production is associative up to isomorphism ((CMR+96)) since it is based on the construction of coproducts. In the following, we denote a parallel production $((...(\hat{p}_1 + \hat{p}_2) + ...) + \hat{p}_n)$ by $\sum_1^n \hat{p}_i$ which simplifies the syntax but does not imply that coproducts are unique.

[8] Since we are not interested in classical language aspects here, we do not distinguish terminal and non-terminal graphs.

In this paper, we investigated distributed graph transformation which is a kind of structured transformation of structured graphs. Parallel graph transformation can be regarded to be a special case of distributed graph transformation where the transformed graphs are non-distributed, i.e. unstructured. (See (Tae96) for further details in this direction.)

The distribution operations "split" and "join" presented in (EBHL88) can be simulated in distributed graph transformation as presented in sections 3 and 4. Both concepts can be formalized by using the distributed graph transformation approach with non-injective network morphisms.

In (Koc97), distributed graph transformation is extended by the concept of application conditions for distributed productions to express additional network, local and synchronization condition.

All distribution concepts presented (the just mentioned application conditions included) are formalized on the basis of the double-pushout approach to graph transformation as well as diagrams and natural transformations. Since the main concepts are characterized in a purely categorical way they may be generalized to high-level replacement systems (EHKP91) exchanging the category **GRAPH** of graphs by another cocomplete category.

Since we have shown that distributed graph transformation fits into the *high-level replacement* framework, it is likely that results for high-level replacement systems concerning concepts such as parallelism, concurrency and embedding can be carried over to distributed graph transformation.

References

A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic Approaches to Graph Transformation I: Basic Concepts and Double Pushout Approach. In G. Rozenberg (ed.), *The Handbook of Graph Grammars, Volume 1: Foundations*. World Scientific, 1996. To appear. Preprint available as technical report TR-96-17, University of Pisa, Department of Computer Science.

P. Degano and U. Montanari. A model of distributed systems based on graph rewriting. *Journal of the ACM*, 34(2):411–449, 1987.

H. Ehrig, P. Böhm, U. Hummert, and M. Löwe. Distributed parallelism of graph transformation. In *13th Int. Workshop on Graph Theoretic Concepts in Computer Science, LNCS 314*, pages 1–19, Berlin, 1988. Springer Verlag.

H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. From Graph Grammars to High Level Replacement Systems. In *4th Int. Workshop on Graph Grammars and their Application to Computer Science, LNCS 532*, pages 269–291. Springer Verlag, 1991.

H. Ehrig. Introduction to the Algebraic Theory of Graph Grammars. In V. Claus, H. Ehrig, and G. Rozenberg (eds.), *1st Graph Grammar Workshop, Lecture Notes in Computer Science 73*, pages 1–69. Springer Verlag, 1979.

H. Ehrig, H.-J. Kreowski, and G. Rozenberg (eds.). *4th International Workshop on Graph Grammars and Their Application to Computer Science*. Springer Verlag, 1991. Lecture Notes in Computer Science 532.

H. Ehrig and M. Löwe. Categorical principles, techniques and results for high-level replacement systems in computer science. *Applied Categorical Structures*, 1(1):21–50, 1993.

H. Ehrig, M. Pfender, and H. Schneider. Graph grammars: an algebraic approach. In *14th Annual IEEE Symposium on Switching and Automata Theory*, pages 167–180. IEEE, 1973.

H. Herrlich and G. Strecker. *Category Theory*. Allyn and Bacon, Rockleigh, New Jersey, 1973.

D. Janssens and G. Rozenberg. Structured Transformations and Computation Graphs for Actor Grammars. In Ehrig et al. (EKR91), pages 446–460. Lecture Notes in Computer Science 532.

D. Kips and G. Heidenreich. Project Flow Graphs – A Meta-Model to support Quality Assurance in Software-Engineering. In *Proc. of IEPM'95*. 1995. to appear.

S. Kaplan, J. Loyall, and S. Goering. Specifying Concurrent Languages and Systems with Δ-Grammars. In Ehrig et al. (EKR91), pages 475–489. Lecture Notes in Computer Science 532.

M. Koch. Bedingte verteilte Graphtransformation und ihre Anwendung auf verteilte Transaktionen. to appear as technical report, TU Berlin, 1997.

M. Korff. Single Pushout Transformations of Equationally Defined Graph Structures with Applications to Actor Systems. In *Proc. Graph Grammar Workshop Dagstuhl 93*, pages 234–247. Springer Verlag, 1994. Lecture Notes in Computer Science 776.

S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer, New York, 1971.

P. Pepper and Wirsing, M., et al. KORSO – A Methodology for the Development of Correct Software. In M. Broy and S. Jähnichen (eds.), KORSO , *Correct Software by Formal Methods, LNCS*. Springer, 1994.

G. Schied. *Über Graphgrammatiken, eine Spezifikationsmethode für Programmiersprachen und verteilte Regelsysteme*. Arbeitsberichte des Institus für mathematische Maschinen und Datenverarbeitung (Informatik), University of Erlangen, 1992.

H.-J. Schneider. On categorical graph grammars integrating structural transformation and operations on labels. *TCS*, 109:257 – 274, 1993.

G. Taentzer. *Parallel and Distributed Graph Transformation: Formal Description and Application to Communication-Based Systems*. PhD thesis, TU Berlin, 1996. Shaker Verlag.

G. Taentzer. Distributed Graph Transformation with Rule-Based Split and Join Operations. to appear as technical report, TU Berlin, 1997.

A. Wagner. Vergleich von High-Level-Replacement-Systemen basierend auf dem Double- bzw. Single-Pushout-Ansatz. Master's thesis, Technical University of Berlin, Dep. of Comp. Sci., 1993.

D. Wolz. Tool Design for Structuring Mechanisms for Algebraic Specification Languages with Initial Semantics. In *11th ADT Workshop & 8th Compass General Meeting Selected Papers*. Springer, LNCS, 1995. to appear.