Multi-Amalgamation in Adhesive Categories

Ulrike Golas[1], Hartmut Ehrig[1], and Annegret Habel[2]

[1] Technische Universität Berlin, Germany
{ugolas,ehrig}@cs.tu-berlin.de
[2] Universität Oldenburg, Germany
annegret.habel@informatik.uni-oldenburg.de

Abstract. Amalgamation is a well-known concept for graph transformations in order to model synchronized parallelism of rules with shared subrules and corresponding transformations. This concept is especially important for an adequate formalization of the operational semantics of statecharts and other visual modeling languages, where typed attributed graphs are used for multiple rules with general application conditions. However, the theory of amalgamation for the double pushout approach has been developed up to now only on a set-theoretical basis for pairs of standard graph rules without any application conditions.

For this reason, we present the theory of amalgamation in this paper in the framework of adhesive categories for a bundle of rules with (nested) application conditions. In fact, it is also valid for weak adhesive HLR categories. The main result is the Multi-Amalgamation Theorem, which generalizes the well-known Parallelism and Amalgamation Theorems to the case of multiple synchronized parallelism.

The constructions are illustrated by a small running example. A more complex case study for the operational semantics of statecharts based on multi-amalgamation is presented in a separate paper.

1 Introduction

1.1 Historical Background of Amalgamation

The concepts of adhesive [1] and (weak) adhesive high-level replacement (HLR) [2] categories have been a break-through for the double pushout approach of algebraic graph transformations [3]. Almost all main results could be formulated and proven in these categorical frameworks and instantiated to a large variety of HLR systems, including different kinds of graph and Petri net transformation systems [2]. These main results include the Local Church–Rosser, Parallelism, and Concurrency Theorems, the Embedding and Extension Theorem, completeness of critical pairs, and the Local Confluence Theorem.

However, at least one main result is missing up to now. The Amalgamation Theorem in [4] has been developed only on a set-theoretical basis for a pair of standard graph rules without application conditions. In [4], the Parallelism Theorem of [5] is generalized to the Amalgamation Theorem, where the assumption

H. Ehrig et al. (Eds.): ICGT 2010, LNCS 6372, pp. 346–361, 2010.

of parallel independence is dropped and pure parallelism is generalized to synchronized parallelism. The synchronization of two rules p_1 and p_2 is expressed by a common subrule p_0, which we call *kernel rule* in this paper. The subrule concept is formalized by a rule morphism $s_i : p_0 \to p_i$, called *kernel morphism* in this paper, based on pullbacks and a pushout complement property. p_1 and p_2 can be glued along p_0 leading to an amalgamated rule $\tilde{p} = p_1 +_{p_0} p_2$. The Amalgamation Theorem states that each amalgamable pair of direct transformations $G \overset{p_i,m_i}{\Longrightarrow} G_i (i = 1, 2)$ via p_1 and p_2 leads to an amalgamated transformation $G \overset{\tilde{p},\tilde{m}}{\Longrightarrow} H$ via $\tilde{p}$, and vice versa yielding a bijective correspondence.

Moreover, the Complement Rule Theorem in [4] allows to construct a complement rule $\overline{p}$ of a kernel morphism $s : p_0 \to p$ leading to a concurrent rule $p_0 *_E \overline{p}$ which is equal to p. Now the Concurrency Theorem allows to decompose each amalgamated transformation $G \overset{\tilde{p},\tilde{m}}{\Longrightarrow} H$ into sequences $G \overset{p_i}{\Longrightarrow} G_i \overset{q_i}{\Longrightarrow} H$ for $i = 1, 2$ and vice versa, where q_i is the complement rule of $t_i : p_i \to \tilde{p}$.

The concepts of amalgamation are applied to communication based systems and visual languages in [4, 6, 7, 8] and transferred to the single-pushout approach of graph transformation in [9]. Other approaches dealing with the problem of similar parallel actions use a collection operator [10] or multi-objects for cloning the complete matches [11]. In [12], an approach based on nested graph predicates combined with amalgamation is introduced which define a relationship between rules and matches. While nesting extends the expressiveness of these transformations, it is quite complicated to write and understand these predicates and, as for the other approaches, it seems to be difficult to relate or integrate them to the theoretical results for graph transformation.

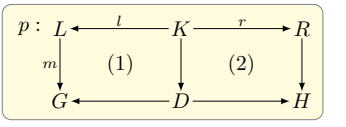
1.2 The Aim of This Paper

The concept of amalgamation plays a key role in the application of parallel graph transformation to communication-based systems [7] and in the modeling of the operational semantics for visual languages [8]. However, in most of these applications we need amalgamation for n rules, called multi-amalgamation, based not only on standard graph rules, but on different kinds of typed and attributed graph rules including (nested) application conditions.

The main idea of this paper is to fill this gap between theory and applications. For this purpose, we have developed the theory of multi-amalgamation for adhesive and adhesive HLR systems based on rules with application conditions. This allows to instantiate the theory to a large variety of graphs and corresponding graph transformation systems and, using weak adhesive HLR categories, also to typed attributed graph transformation systems [2]. A complex case study for the operational semantics of statecharts based on typed attributed graphs and multi-amalgamation is presented in [13]. For simplicity and due to space limitations, we present the theory in this paper for adhesive categories, while weak adhesive HLR categories are considered in [14].

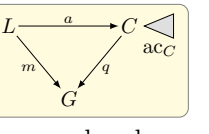
1.3 Review of Basic Notions

The basic idea of adhesive categories [1] is to have a category with pushouts along monomorphisms and pullbacks satisfying the van Kampen property. Intuitively, this means that pushouts along monomorphisms and pullbacks are compatible with each other. This holds for sets and various kinds of graphs (see [1, 2]), including the standard category of graphs which is used as a running example in this paper.



In the double pushout approach to graph transformation, a rule is given by a span $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ with objects L, K, and R, called left-hand side, interface, and right-hand side, respectively, and monomorphisms l and r. An application of such a rule to a graph G via a match $m : L \to G$ is constructed as two gluings (1) and (2), which are pushouts in the corresponding graph category, leading to a direct transformation $G \overset{p,m}{\Longrightarrow} H$.

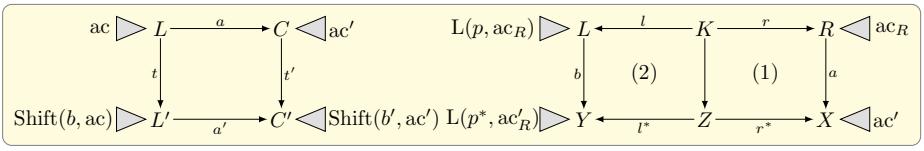
An important extension is the use of rules with suitable application conditions. These include positive application conditions of the form $\exists a$ for a morphism $a : L \to C$, demanding a certain structure in addition to L, and also negative application conditions $\neg\exists a$, forbidding such a structure. A match $m : L \to G$ satisfies $\exists a$ ($\neg\exists a$) if there is a (no) monomorphism $q : C \to G$ satisfying $q \circ a = m$. In more detail, we use nested application conditions [15], short application conditions. In particular, true is an application condition which is always satisfied. For a basic application condition $\exists(a, \mathrm{ac}_C)$ on L with an application condition ac_C on C, in addition to the existence of q it is required that q satisfies ac_C. In particular, we have $\exists a = \exists(a, \mathrm{true})$ for $ac_C = \mathrm{true}$. In general, we write $m \models \exists(a, \mathrm{ac}_C)$ if m satisfies $\exists(a, \mathrm{ac}_C)$, and application conditions are closed under boolean operations. Moreover, $\mathrm{ac}_C \cong \mathrm{ac}'_C$ denotes the semantical equivalence of ac_C and ac'_C on C.



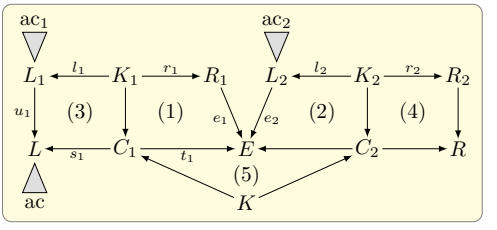
In this paper we consider rules of the form $p = (L \xleftarrow{l} K \xrightarrow{r} R, \mathrm{ac})$, where $(L \xleftarrow{l} K \xrightarrow{r} R)$ is a (plain) rule and ac is an application condition on L. In order to handle rules with application conditions there are two important concepts, called the shift of application conditions over morphisms and rules ([15, 16]):

1. Given an application condition ac_L on L and a morphism $t : L \to L'$ then there is an application condition $\mathrm{Shift}(t, \mathrm{ac}_L)$ on L' such that for all $m' : L' \to G$ holds: $m' \models \mathrm{Shift}(t, \mathrm{ac}_L) \iff m = m' \circ t \models \mathrm{ac}_L$. For a basic application condition $\mathrm{ac} = \exists(a, \mathrm{ac}')$ we define $\mathrm{Shift}(t, \mathrm{ac}) = \vee_{(a',t') \in \mathcal{F}} \exists(a', \mathrm{Shift}(t', \mathrm{ac}'))$ with $\mathcal{F} = \{(a', t') \mid (a', t')\text{jointly epimorphic}, t' \in \mathcal{M}, t' \circ a = a' \circ t\}$,
2. Given a plain rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ and an application condition ac_R on R then there is an application condition $\mathrm{L}(p, \mathrm{ac}_R)$ on L such that for all transformations $G \overset{p,m,n}{\Longrightarrow} H$ with match m and comatch n holds: $m \models \mathrm{L}(p, \mathrm{ac}_R) \iff n \models \mathrm{ac}_R$. For a basic application condition $\mathrm{ac}_R = \exists(a, \mathrm{ac}'_R)$ we define $\mathrm{L}(p, \mathrm{ac}_R) = \exists(b, \mathrm{L}(p^*, \mathrm{ac}'_R))$ if $a \circ r$ has a pushout complement (1)

and $p^* = (Y \xleftarrow{l^*} Z \xrightarrow{r^*} X)$ is the derived rule by constructing pushout (2), and $\mathrm{L}(p, \exists(a, \mathrm{ac}'_R)) =$ false otherwise. Vice versa, there is also a construction $\mathrm{R}(p, \mathrm{ac}_L)$ shifting an application condition ac_L on L over p to R.



One of the main results for graph transformation is the Concurrency Theorem, which is concerned with the execution of transformations which may be sequentially dependent. Given an arbitrary sequence $G \overset{p_1,m_1}{\Longrightarrow} H \overset{p_2,m_2}{\Longrightarrow} G'$ of direct transformations it is possible to construct an E-concurrent rule $p_1 *_E p_2 = (L \leftarrow K \rightarrow R, \mathrm{ac})$ by pushouts (1), (2), (3), (4), and pullback (5). The object E is an overlap of R_1 and L_2, where the two overlapping morphisms have to be in a class $\mathcal{E}'$ of pairs of morphisms with the same codomain. The construction of the concurrent application condition $\mathrm{ac} = \mathrm{Shift}(u_1, \mathrm{ac}_1) \wedge \mathrm{L}(p^*, \mathrm{Shift}(e_2, \mathrm{ac}_2))$ and $p^* = (L \xleftarrow{s_1} C_1 \xrightarrow{t_1} E)$ is again based on the two shift constructions. The Concurrency Theorem states that for the transformation $G \overset{p_1,m_1}{\Longrightarrow} H \overset{p_2,m_2}{\Longrightarrow} G'$ the E-concurrent rule $p_1 *_E p_2$ allows us to construct a direct transformation $G \overset{p_1 *_E p_2}{\Longrightarrow} G'$ via $p_1 *_E p_2$, and vice versa, each direct transformation $p_1 *_E p_2$ can be sequentialized.



1.4 General Assumptions

In this paper we assume to have an adhesive category [1] with binary coproducts, epi-mono-factorization, and initial pushouts [2]. We consider rules with (nested) application conditions [15] as explained above. In the following, a *bundle* represents a family of morphisms or transformation steps with the same domain, which means that a bundle of things always starts at the same object. Note that the theory is also valid for weak adhesive HLR categories with a suitable class $\mathcal{M}$ of monomorphisms [2, 14].

1.5 Organization of This Paper

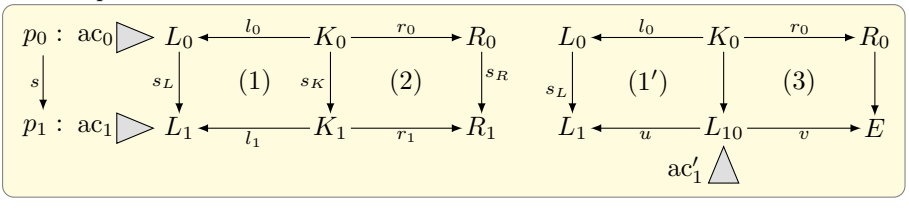
This paper is organized as follows: in Section 2, we introduce kernel rules, multi rules, and kernel morphisms leading to the Complement Rule Theorem as first main result. In Section 3, we construct multi-amalgamated rules and transformations and show as second main result the Multi-Amalgamation Theorem. In Section 4, we present a summary of our results and discuss future work. In the main part of the paper, we mainly give proof ideas, the full proofs can be found in [14].

2 Decomposition of Direct Transformations

In this section, we show how to decompose a direct transformation in adhesive categories into transformations via a kernel and a complement rule leading to the Complement Rule Theorem.

A kernel morphism describes how a smaller rule, the kernel rule, is embedded into a larger rule, the multi rule, which has its name because it can be applied multiple times for a given kernel rule match as described in Section 3. We need some more technical preconditions to make sure that the embeddings of the L-, K-, and R-components as well as the application conditions are consistent and allow to construct a complement rule.

Definition 1 (Kernel morphism). *Given rules* $p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, \mathrm{ac}_0)$ *and* $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \mathrm{ac}_1)$*, a* kernel morphism $s : p_0 \to p_1$*,* $s = (s_L, s_K, s_R)$ *consists of monomorphisms* $s_L : L_0 \to L_1$*,* $s_K : K_0 \to K_1$*, and* $s_R : R_0 \to R_1$ *such that in the following diagram* (1) *and* (2) *are pullbacks,* (1) *has a pushout complement* (1′) *for* $s_L \circ l_0$*, and* ac_0 *and* ac_1 *are* complement-compatible w.r.t. s*, i.e. given pushout* (3) *then* $\mathrm{ac}_1 \cong \mathrm{Shift}(s_L, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(v, \mathrm{ac}_1'))$ *for some* ac_1' *on* L_{10} *and* $p_1^* = (L_1 \xleftarrow{u} L_{10} \xrightarrow{v} E_1)$*. In this case,* p_0 *is called* kernel rule *and* p_1 multi rule*.*



Remark 1. The complement-compatibility of the application conditions makes sure that there is a decomposition of ac_1 into parts on L_0 and L_{10}, where the latter ones are used later for the application conditions of the complement rule. L_{10} represents in addition to K_0 all these elements that are new in the multi rule and thus have to be present in the complement rule. This ensures that the complement rule is applicable after the kernel rule if and only if the multi rule can be applied. Otherwise, ac_1 uses combined elements from the kernel and complement rules such that the multi rule cannot be decomposed.

Example 1. To explain the concept of amalgamation, in our example we model a small transformation system for switching the direction of edges in labeled graphs, where we only have different labels for edges – black and dotted edges. The kernel rule p_0 is depicted in the top of Fig. 1. It selects a node with a black loop, deletes this loop, and adds a dotted loop, all of this if no dotted loop is already present. The matches are defined by the numbers at the nodes and can be induced for the edges by their position.

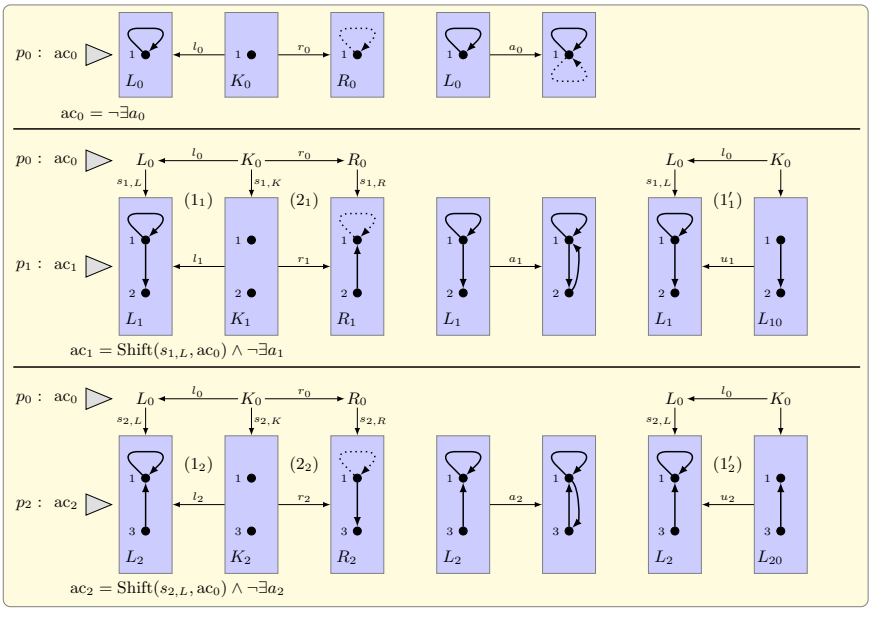


Fig. 1. The kernel rule p_0 and the multi rules p_1 and p_2

In the middle and bottom of Fig. 1, two multi rules p_1 and p_2 are shown, which extend the rule p_0 and in addition reverse an edge if no backward edge is present. They also inherit the application condition of p_0 forbidding a dotted loop at the selected node.

There is a kernel morphism $s_1 : p_0 \to p_1$ as shown in the top of Fig. 1 with pullbacks (1_1) and (2_1), and pushout complement $(1'_1)$. For the application conditions, $\mathrm{ac}_1 = \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \neg\exists a_1 \cong \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(v_1, \neg\exists a'_1))$ with a'_1 as shown in Fig. 2. We have that $\mathrm{Shift}(v_1, \neg\exists a'_1) = \neg\exists a_{11}$, because square $(*)$ is the only possible commuting square leading to a_{11}, b_{11} jointly surjective and b_{11} injective. Moreover, $\mathrm{L}(p_1^*, \neg\exists a_{11}) = \neg\exists a_1$ as shown by the two pushout squares (PO_1) and (PO_2) in Fig. 2. Thus $\mathrm{ac}'_1 = \neg\exists a'_1$, and ac_0 and ac_1 are complement compatible.

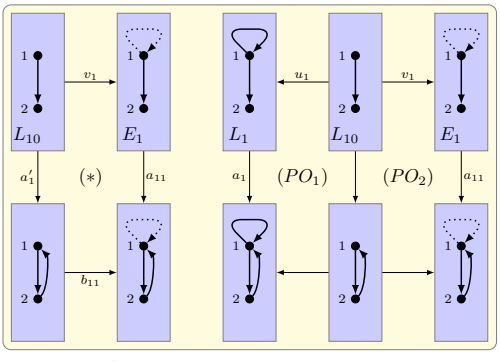
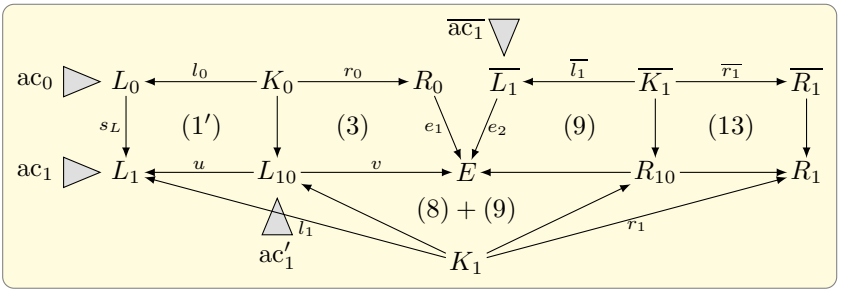


Fig. 2. Application condition construction

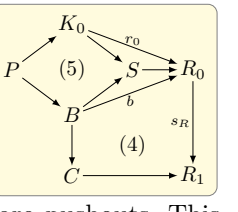
Similarly, there is a kernel morphism $s_2 : p_0 \to p_2$ as shown in the bottom of Fig. 1 with pullbacks (1_2) and (2_2), pushout complement $(1'_2)$, and ac_0 and ac_2 are complement compatible.

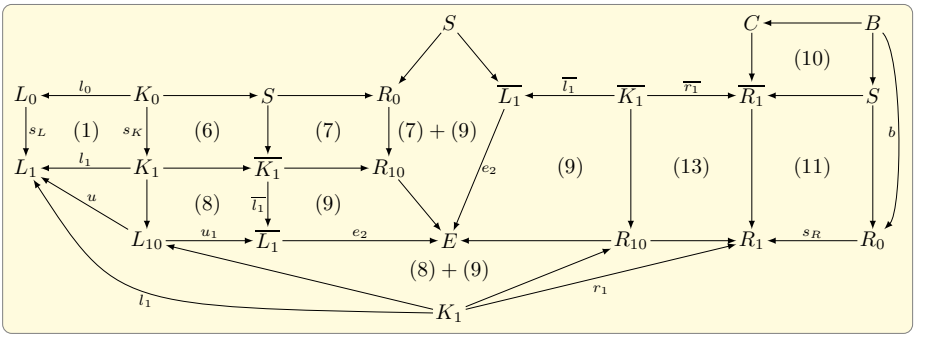
For a given kernel morphism, the complement rule is the remainder of the multi rule after the application of the kernel rule, i.e. it describes what the multi rule does in addition to the kernel rule.

Theorem 1 (Existence of complement rule). *Given rules* $p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, \mathrm{ac}_0)$ *and* $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \mathrm{ac}_1)$*, and a kernel morphism* $s : p_0 \to p_1$ *then there exists a rule* $\overline{p_1} = (\overline{L_1} \xleftarrow{\overline{l_1}} \overline{K_1} \xrightarrow{\overline{r_1}} \overline{R_1}, \overline{\mathrm{ac}_1})$ *and a jointly epimorphic cospan* $R_0 \xrightarrow{e_1} E \xleftarrow{e_2} \overline{L_1}$ *such that the* E*-concurrent rule* $p_0 *_E \overline{p_1}$ *exists and* $p_1 = p_0 *_E \overline{p_1}$*.*



Proof idea. We consider the construction without application conditions. First, S is constructed as follows: we construct the initial pushout (4) over s_R, P as the pullback object of r_0 and b, and then the pushout (5). After the construction of different pushouts and pullbacks and using various properties of adhesive categories, we obtain the diagram below where all squares (6) to (11) and (13) are pushouts. This leads to the required rule $\overline{p_1} = (\overline{L_1} \xleftarrow{\overline{l_1}} \overline{K_1} \xrightarrow{\overline{r_1}} \overline{R_1})$ and $p_1 = p_0 *_E \overline{p_1}$ for rules without application conditions.





For the application conditions, suppose $\mathrm{ac}_1 \cong \mathrm{Shift}(s_L, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(v, \mathrm{ac}_1'))$ for $p_1^* = (L_1 \xleftarrow{u} L_{10} \xrightarrow{v} E)$ with $v = e_2 \circ u_1$ and ac_1' on L_{10}. Now define $\overline{\mathrm{ac}_1} = \mathrm{Shift}(u_1, \mathrm{ac}_1')$, which is an application condition on $\overline{L_1}$.

We have to show that $(p_1, \mathrm{ac}_{p_0 *_E \overline{p_1}}) \cong (p_1, \mathrm{ac}_1)$. By construction of the E-concurrent rule we have that $\mathrm{ac}_{p_0 *_E \overline{p_1}} \cong \mathrm{Shift}(s_L, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(e_2, \overline{\mathrm{ac}_1})) \cong$

$\mathrm{Shift}(s_L, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(e_2, \mathrm{Shift}(u_1, \mathrm{ac}_1'))) \cong \mathrm{Shift}(s_L, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(e_2 \circ u_1, \mathrm{ac}_1')) \cong \mathrm{Shift}(s_L, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(v, \mathrm{ac}_1')) \cong \mathrm{ac}_1$. □

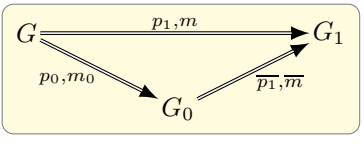
Remark 2. Note, that by construction the interface K_0 of the kernel rule has to be preserved in the complement rule. The construction of $\overline{p_1}$ is not unique w.r.t. the property $p_1 = p_0 *_E \overline{p_1}$, since other choices for S with monomorphisms from K_0 and B also lead to a well-defined construction. In particular, one could choose $S = R_0$ leading to $\overline{p_1} = E \leftarrow R_{10} \rightarrow R_1$. Our choice represents a smallest possible complement, which should be preferred in most application areas.

Definition 2 (Complement rule). *Given rules* $p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, \mathrm{ac}_0)$ *and* $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \mathrm{ac}_1)$*, and a kernel morphism* $s : p_0 \to p_1$ *then the rule* $\overline{p_1} = (\overline{L_1} \xleftarrow{\overline{l_1}} \overline{K_1} \xrightarrow{\overline{r_1}} \overline{R_1}, \overline{\mathrm{ac}_1})$ *constructed in Thm. 1 is called* complement rule *(of* s*).*

Example 2. Consider the kernel morphism s_1 depicted in Fig. 1. Using the construction in Thm. 1 we obtain the diagrams in Fig. 3 leading to the complement rule in the top row in Fig. 4 with the application condition $\overline{\mathrm{ac}_1} = \neg\exists\overline{a_1}$. Similarly, we obtain a complement rule for the kernel morphism $s_2 : p_0 \to p_2$ in Fig. 1, which is depicted in the bottom row of Fig. 4.

Each direct transformation via a multi rule can be decomposed into a direct transformation via the kernel rule followed by a direct transformation via the complement rule.

Fact 1. *Given rules* $p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, \mathrm{ac}_0)$ *and* $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \mathrm{ac}_1)$*, a kernel morphism* $s : p_0 \to p_1$*, and a direct transformation* $t : G \xRightarrow{p_1, m} G_1$ *then* t *can be decomposed into the transformation* $G \xRightarrow{p_0, m_0} G_0 \xRightarrow{\overline{p_1}, \overline{m}} G_1$ *with* $m_0 = m \circ s_L$ *where* $\overline{p_1}$ *is the complement rule of* s*.*



Proof. We have that $p_1 \cong p_0 *_E \overline{p_1}$. The analysis part of the Concurrency Theorem [16] now implies the decomposition into $G \xRightarrow{p_0, m_0} G_0 \xRightarrow{\overline{p_1}, \overline{m}} G_1$ with $m_0 = m \circ s_L$. □

3 Multi-Amalgamation

In [4], an Amalgamation Theorem for a pair of graph rules without application conditions has been developed. It can be seen as a generalization of the Parallelism Theorem [5], where the assumption of parallel independence is dropped and pure parallelism is generalized to synchronized parallelism. In this section, we present an Amalgamation Theorem for a bundle of rules with application conditions, called Multi-Amalgamation Theorem, over objects in an adhesive category.

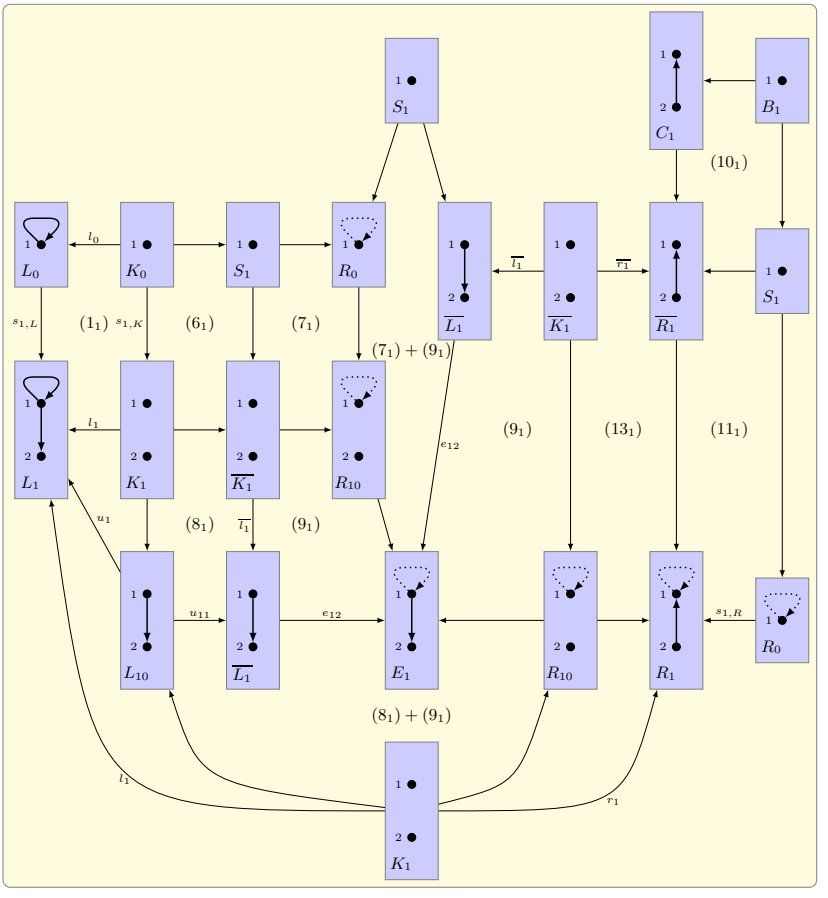


Fig. 3. The construction of the complement rule for the kernel morphism s_1

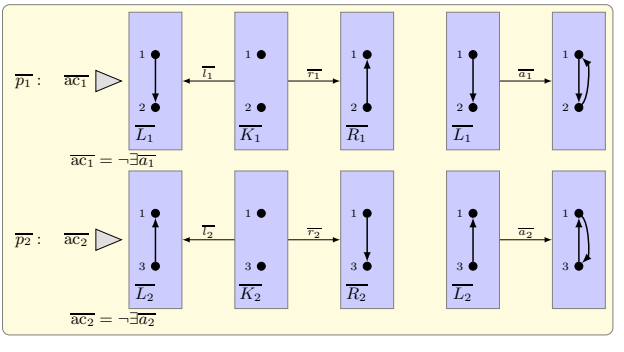
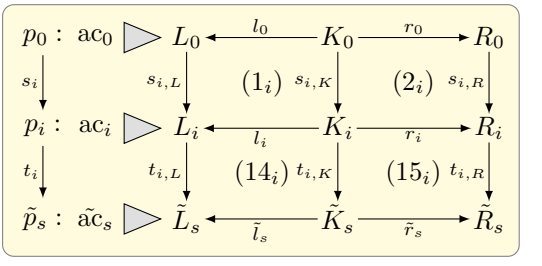


Fig. 4. The complement rules for the kernel morphisms

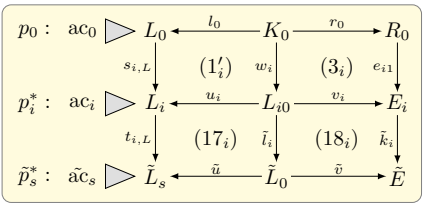
We consider not only single kernel morphisms, but bundles of them over a fixed kernel rule. Then we can combine the multi rules of such a bundle to an amalgamated rule by gluing them along the common kernel rule.

Definition 3 (Multi-amalgamated rule). *Given rules* $p_i = (L_i \xleftarrow{l_i} K_i \xrightarrow{r_i} R_i, \mathrm{ac}_i)$ *for* $i = 0, \ldots, n$ *and a bundle of kernel morphisms* $s = (s_i : p_0 \to p_i)_{i=1,\ldots,n}$*, then the* (multi-)amalgamated rule $\tilde{p}_s = (\tilde{L}_s \xleftarrow{\tilde{l}_s} \tilde{K}_s \xrightarrow{\tilde{r}_s} \tilde{R}_s, \tilde{\mathrm{ac}}_s)$ *is constructed as the componentwise colimit of the kernel morphisms:*

- $\tilde{L}_s = Col((s_{i,L})_{i=1,\ldots,n})$,
- $\tilde{K}_s = Col((s_{i,K})_{i=1,\ldots,n})$,
- $\tilde{R}_s = Col((s_{i,R})_{i=1,\ldots,n})$,
- $\tilde{l}_s$ *and* $\tilde{r}_s$ *are induced by* $(t_{i,L} \circ l_i)_{i=0,\ldots,n}$ *and* $(t_{i,R} \circ r_i)_{0=1,\ldots,n}$*, respectively,*
- $\tilde{\mathrm{ac}}_s = \bigwedge_{i=1,\ldots,n} \mathrm{Shift}(t_{i,L}, \mathrm{ac}_i)$.



Fact 2. *The amalgamated rule is well-defined and we have kernel morphisms* $t_i = (t_{i,L}, t_{i,K}, t_{i,R}) : p_i \to \tilde{p}_s$ *for* $i = 0, \ldots, n$.



Proof idea. The colimit of a bundle of n morphisms can be constructed by iterated pushout constructions, which means that we only have to require pushouts over monomorphisms. Since pushouts are closed under monomorphisms, the iterated pushout construction leads to t being a monomorphism. In addition, it can be shown by induction that that (14_i) resp. $(14_i) + (1_i)$ and (15_i) resp. $(15_i) + (2_i)$ are pullbacks, and (14_i) resp. $(14_i) + (1_i)$ has a pushout complement (17_i) resp. $(17_i) + (1'_i)$ for $t_{i,L} \circ l_i$.

Since ac_0 and ac_i are complement-compatible for all i we have that $\mathrm{ac}_i \cong \mathrm{Shift}(s_{i,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_i^*, \mathrm{Shift}(v_i, \mathrm{ac}'_i))$. For any ac'_i, it holds that $\mathrm{Shift}(t_{i,L}, \mathrm{L}(p_i^*, \mathrm{Shift}(v_i, \mathrm{ac}'_i)))) \cong \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{k}_i \circ v_i, \mathrm{ac}'_i)) \cong \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{v}, \mathrm{Shift}(\tilde{l}_i, \mathrm{ac}'_i)))$, since all squares are pushouts by pushout-pullback decomposition and the uniqueness of pushout complements. Define $\mathrm{ac}_i^* := \mathrm{Shift}(\tilde{l}_i, \mathrm{ac}'_i)$ as an application condition on $\tilde{L}_0$. It follows that $\tilde{\mathrm{ac}}_s = \bigwedge_{i=1,\ldots,n} \mathrm{Shift}(t_{i,L}, \mathrm{ac}_i) \cong \bigwedge_{i=1,\ldots,n} (\mathrm{Shift}(t_{i,L} \circ s_{i,L}, \mathrm{ac}_0) \wedge \mathrm{Shift}(t_{i,L}, \mathrm{L}(p_i^*, \mathrm{Shift}(v_i, \mathrm{ac}'_i)))) \cong \mathrm{Shift}(t_{0,L}, \mathrm{ac}_0) \wedge \bigwedge_{i=1,\ldots,n} \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{v}, \mathrm{ac}_i^*))$.

For $i = 0$ define $\mathrm{ac}'_{s0} = \bigwedge_{j=1,\ldots,n} ac_j^*$, and hence $\tilde{\mathrm{ac}}_s = \mathrm{Shift}(t_{0,L}, \mathrm{ac}_0) \wedge \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{v}, \mathrm{ac}'_{s0}))$ implies the complement-compatibility of ac_0 and $\tilde{\mathrm{ac}}_s$. For $i > 0$, we have that $\mathrm{Shift}(t_{0,L}, \mathrm{ac}_0) \wedge \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{v}, \mathrm{ac}_i^*)) \cong \mathrm{Shift}(t_{i,L}, \mathrm{ac}_i)$. Define $\mathrm{ac}'_{si} = \bigwedge_{j=1,\ldots,n \setminus i} ac_j^*$, and hence $\tilde{\mathrm{ac}}_s = \mathrm{Shift}(t_{i,L}, \mathrm{ac}_i) \wedge \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{v}, \mathrm{ac}'_{si}))$ implies the complement-compatibility of ac_i and $\tilde{\mathrm{ac}}_s$. □

The application of an amalgamated rule yields an amalgamated transformation.

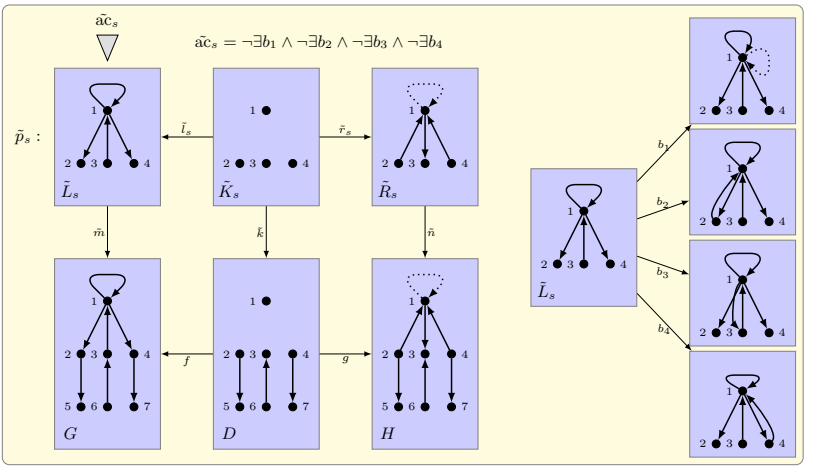


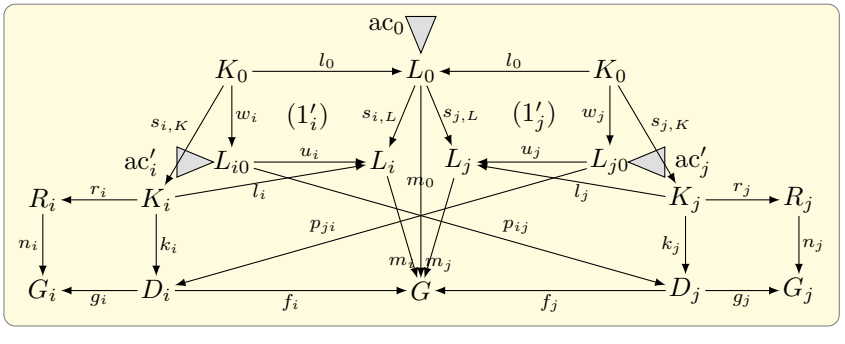
Fig. 5. An amalgamated transformation

Definition 4 (Amalgamated transformation). *The application of an amalgamated rule to a graph G is called an* amalgamated transformation.

Example 3. Consider the bundle $s = (s_1, s_2, s_3 = s_1)$ of the kernel morphisms depicted in Fig. 1. The corresponding amalgamated rule $\tilde{p}_s$ is shown in the top row of Fig. 5. This amalgamated rule can be applied to the graph G leading to the amalgamated transformation depicted in Fig. 5, where the application condition $\tilde{ac}_s$ is obviously fulfilled by the match $\tilde{m}$.

If we have a bundle of direct transformations of a graph G, where for each transformation one of the multi rules is applied, we want to analyze if the amalgamated rule is applicable to G combining all the single transformation steps. These transformations are compatible, i.e. multi-amalgamable, if the matches agree on the kernel rules, and are independent outside.

Definition 5 (Multi-amalgamable). *Given a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,\dots,n}$, a bundle of direct transformations steps $(G \overset{p_i,m_i}{\Longrightarrow} G_i)_{i=1,\dots n}$ is* s-multi-amalgamable*, or short s-amalgamable, if*



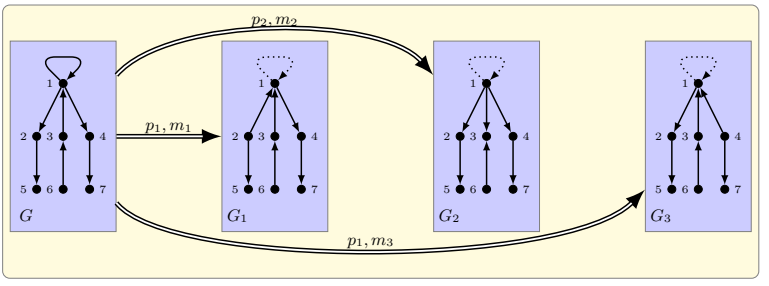
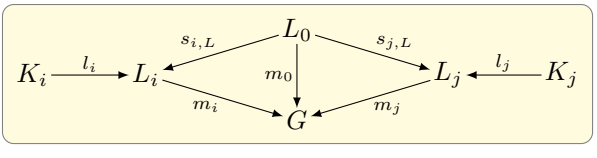


Fig. 6. An s-amalgamable bundle of direct transformations

- *it has* consistent matches, *i.e.* $m_i \circ s_{i,L} = m_j \circ s_{j,L} =: m_0$ *for all* $i, j = 1, \ldots, n$ *and*
- *it has* weakly independent matches, *i.e. for all* $i \neq j$ *consider the pushout complements* $(1'_i)$ *and* $(1'_j)$*, and then there exist morphisms* $p_{ij} : L_{i0} \to D_j$ *and* $p_{ji} : L_{j0} \to D_i$ *such that* $f_j \circ p_{ij} = m_i \circ u_i$, $f_i \circ p_{ji} = m_j \circ u_j$, $g_j \circ p_{ij} \models \mathrm{ac}'_i$, *and* $g_i \circ p_{ji} \models \mathrm{ac}'_j$.

Similar to the characterization of parallel independence in [2] we can give a set-theoretical characterization of weak independence.

Fact 3. *For graphs and other set-based structures, weakly independent matches means that* $m_i(L_i) \cap m_j(L_j) \subseteq m_0(L_0) \cup (m_i(l_i(K_i)) \cap m_j(l_j(K_j)))$ *for all* $i \neq j = 1, \ldots, n$*, i.e. the elements in the intersection of the matches* m_i *and* m_j *are either preserved by both transformations, or are also matched by* m_0.



Proof. We have to proof the equivalence of $m_i(L_i) \cap m_j(L_j) \subseteq m_0(L_0) \cup (m_i(l_i(K_i)) \cap m_j(l_j(K_j)))$ for all $i \neq j = 1, \ldots, n$ with the definition of weakly independent matches.

"$\Leftarrow$" Let $x = m_i(y_i) = m_j(y_j)$, and suppose $x \notin m_0(L_0)$. Since $(1'_i)$ is a pushout we have that $y_i = u_i(z_i) \in u_i(L_{i0} \backslash w_i(K_0))$, and $x = m_i(u_i(z_i)) = f_j(p_i(z_i)) = m_j(y_j)$, and by pushout properties $y_j \in l_j(K_j)$ and $x \in m_j(l_j(K_j))$. Similarly, $x \in m_i(l_i(K_i))$.

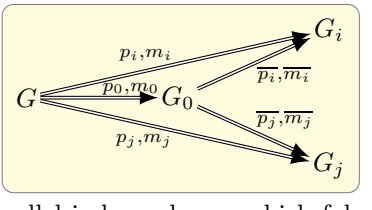
"$\Rightarrow$" For $x \in L_{i0}$, $x = w_i(k)$ define $p_{ij}(x) = k_j(s_{j,K}(k))$, then $f_j(p_{ij}(x)) = f_j(k_j(s_{j,K}(k))) = m_j(l_j(s_{j,K}(k))) = m_j(s_{j,L}(l_0(k))) = m_i(s_{i,L}(l_0(k))) = m_i(n_i(w_i(k))) = m_i(u_i(x))$. Otherwise, $x \notin w_i(K_0)$, i.e. $u_i(x) \notin s_{i,L}(L_0)$, and define $p_{ij}(x) = y$ with $f_j(y) = m_i(u_i(x))$. This y exists, because either $m_i(u_i(x)) \notin m_j(L_j)$ or $m_i(u_i(x)) \in m_j(L_j)$ and then $m_i(u_i(x)) \in m_j(l_j(K_j))$, and in both cases $m_i(u_i(x)) \in f_j(D_j)$. Similarly, we can define p_{ji} with the required property. □

Example 4. Consider the bundle $s = (s_1, s_2, s_3 = s_1)$ of kernel morphisms from Ex. 3. For the graph G given in Fig. 5 we find matches $m_0 : L_0 \to G$, $m_1 : L_1 \to$

G, $m_2 : L_2 \to G$, and $m_3 : L_1 \to G$ mapping all nodes from the left hand side to their corresponding nodes in G, except for m_3 mapping node 2 in L_1 to node 4 in G. For all these matches, the corresponding application conditions are fulfilled and we can apply the rules p_1, p_2, p_1, respectively, leading to the bundle of direct transformations depicted in Fig. 6. This bundle is s-amalgamable, because the matches m_1, m_2, and m_3 agree on the match m_0, and are weakly independent, because they only overlap in m_0.

For an s-amalgamable bundle of direct transformations, each single transformation step can be decomposed into an application of the kernel rule followed by an application of the complement rule. Moreover, all kernel rule applications lead to the same object, and the following applications of the complement rules are parallel independent.

Fact 4. *Given a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,\dots,n}$ and an s-amalgamable bundle of direct transformations $(G \overset{p_i,m_i}{\Longrightarrow} G_i)_{i=1,\dots,n}$ then each direct transformation $G \overset{p_i,m_i}{\Longrightarrow} G_i$ can be decomposed into a transformation $G \overset{p_0,m_0}{\Longrightarrow} G_0 \overset{\overline{p_i},\overline{m_i}}{\Longrightarrow} G_i$. Moreover, the transformations $G_0 \overset{\overline{p_i},\overline{m_i}}{\Longrightarrow} G_i$ are pairwise parallel independent.*

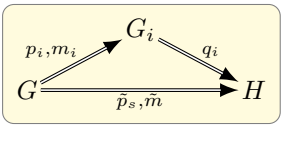


Proof idea. From Fact 1 it follows that each single direct transformation $G \overset{p_i,m_i}{\Longrightarrow} G_i$ can be decomposed into a transformation $G \overset{p_0,m_0^i}{\Longrightarrow} G_0^i \overset{\overline{p_i},\overline{m_i}}{\Longrightarrow} G_i$ with $m_0^i = m_i \circ s_{i,L}$, and since the bundle is s-amalgamable, $m_0 = m_i \circ s_{i,L} = m_0^i$ and $G_0 := G_0^i$ for all $i = 1, \dots, n$. It remains to show the pairwise parallel independence, which follows from consistent and weakly independent matches. □

If a bundle of direct transformations of a graph G is s-amalgamable, then we can apply the amalgamated rule directly to G leading to a parallel execution of all the changes done by the single transformation steps.

Theorem 2 (Multi-Amalgamation). *Consider a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,\dots,n}$.*

1. Synthesis. *Given an s-amalgamable bundle of direct transformations $(G \overset{p_i,m_i}{\Longrightarrow} G_i)_{i=1,\dots,n}$ then there is an amalgamated transformation $G \overset{\tilde{p}_s,\tilde{m}}{\Longrightarrow} H$ and transformations $G_i \overset{q_i}{\Longrightarrow} H$ over the complement rules q_i of the kernel morphisms $t_i : p_i \to \tilde{p}_s$ such that $G \overset{p_i,m_i}{\Longrightarrow} G_i \overset{q_i}{\Longrightarrow} H$ is a decomposition of $G \overset{\tilde{p}_s,\tilde{m}}{\Longrightarrow} H$.*



2. Analysis. *Given an amalgamated transformation* $G \overset{\tilde{p}_s,\tilde{m}}{\Longrightarrow} H$ *then there are* s_i*-related transformations* $G \overset{p_i,m_i}{\Longrightarrow} G_i \overset{q_i}{\Longrightarrow} H$ *for* $i = 1,\ldots,n$ *such that* $G \overset{p_i,m_i}{\Longrightarrow} G_i$ *is* s*-amalgamable.*
3. Bijective Correspondence. *The synthesis and analysis constructions are inverse to each other up to isomorphism.*

Proof idea.

1. We can show that $\tilde{p}_s$ is applicable to G leading to an amalgamated transformation $G \overset{\tilde{p}_s,\tilde{m}}{\Longrightarrow} H$ with $m_i = \tilde{m} \circ t_{i,L}$, where $t_i : p_i \to \tilde{p}_i$ is the kernel morphism constructed in Fact 2. Then we can apply Fact 1 which implies the decomposition of $G \overset{\tilde{p}_s,\tilde{m}}{\Longrightarrow} H$ into $G \overset{p_i,m_i}{\Longrightarrow} G_i \overset{q_i}{\Longrightarrow} H$, where q_i is the complement rule of the kernel morphism t_i.
2. Using the kernel morphisms t_i we obtain transformations $G \overset{p_i,m_i}{\Longrightarrow} G_i \overset{q_i}{\Longrightarrow} H$ from Fact 1 with $m_i = \tilde{m} \circ t_{i,L}$. We have to show that this bundle of transformation is s-amalgamable. Applying again Fact 1 we obtain transformations $G \overset{p_0,m_0^i}{\Longrightarrow} G_0^i \overset{\overline{p_i}}{\Longrightarrow} G_i$ with $m_0^i = m_i \circ s_{i,L}$. It follows that $m_0^i = m_i \circ s_{i,L} = \tilde{m} \circ t_{i,L} \circ s_{i,L} = \tilde{m} \circ t_{0,L} = \tilde{m} \circ t_{j,L} \circ s_{j,L} = m_j \circ s_{j,L}$ and thus we have consistent matches with $m_0 := m_0^i$ well-defined and $G_0 = G_0^i$. Moreover, the matches are weakly independent.
3. Because of the uniqueness of the used constructions, the above constructions are inverse to each other up to isomorphism. □

Remark 3. Note, that q_i can be constructed as the amalgamated rule of the kernel morphisms $(p_{K_0} \to \overline{p_j})_{j \neq i}$, where $p_{K_0} = (K_0 \overset{id_{K_0}}{\longleftarrow} K_0 \overset{id_{K_0}}{\longrightarrow} K_0, \text{true})$ and $\overline{p_j}$ is the complement rule of p_j.

For $n = 2$ and rules without application conditions, the Multi-Amalgamation Theorem specializes to the Amalgamation Theorem in [4]. Moreover, if p_0 is the empty rule, this is the Parallelism Theorem in [16], since the transformations are parallel independent for an empty kernel match.

Example 5. As already stated in Example 4, the transformations $G \overset{p_1,m_1}{\Longrightarrow} G_1$, $G \overset{p_2,m_2}{\Longrightarrow} G_2$, and $G \overset{p_1,m_3}{\Longrightarrow} G_3$ shown in Fig. 6 are s-amalgamable for the bundle $s = (s_1, s_2, s_3 = s_1)$ of kernel morphisms. Applying Fact 4, we can decompose these transformations into a transformation $G \overset{p_0,m_0}{\Longrightarrow} G_0$ followed by transformations $G_0 \overset{\overline{p_1},\overline{m_1}}{\Longrightarrow} G_1$, $G_0 \overset{\overline{p_2},\overline{m_2}}{\Longrightarrow} G_2$, and $G_0 \overset{\overline{p_1},\overline{m_3}}{\Longrightarrow} G_3$ via the complement rules, which are pairwise parallel independent. These transformations are depicted in Fig. 7. Moreover, Thm. 2 implies that we obtain for this bundle of direct transformations an amalgamated transformation $G \overset{\tilde{p}_s,\tilde{m}}{\Longrightarrow} H$, which is the transformation already shown in Fig. 5. Vice versa, the analysis of this amalgamated transformation leads to the s-amalgamable bundle of transformations $G \overset{p_1,m_1}{\Longrightarrow} G_1$, $G \overset{p_2,m_2}{\Longrightarrow} G_2$, and $G \overset{p_1,m_3}{\Longrightarrow} G_3$ from Fig. 6.

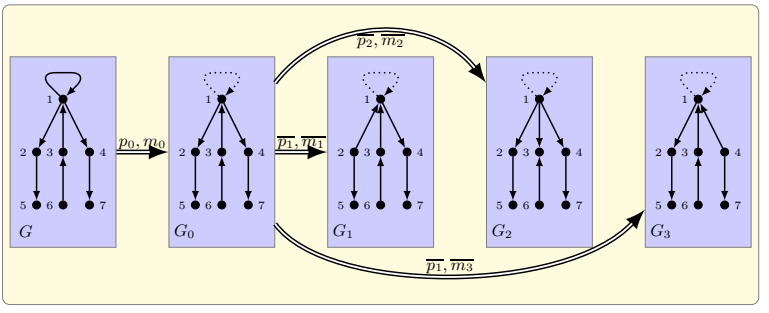


Fig. 7. The decomposition of the s-amalgamable bundle

Extension to Multi-Amalgamation with Maximal Matchings

An important extension of the presented theory is the introduction of interaction schemes and maximal matchings. An interaction scheme defines a bundle of kernel morphisms. In contrast to a concrete bundle, for the application of such an interaction scheme all possible matches for the multi rules are computed that agree on a given kernel match and lead to an amalgamable bundle of transformations. In our example, the interaction scheme $is = \{s_1, s_2\}$ contains the two kernel morphisms from Fig. 1. For the kernel match m_0, the matches m_1, m_2, m_3 are maximal: they are s-amalgamable, and any other match for p_1 or p_2 that agrees an m_0 would hold only already matched elements. This technique is very useful for the definition of the semantics of visual languages. For our example concerning statecharts [13], an unknown number of state transitions triggered by the same event, which is highly dependent on the actual system state, can be handled in parallel.

4 Conclusion

In this paper, we have generalized the theory of amalgamation in [4] to multi-amalgamation in adhesive categories. More precisely, the Complement Rule and Amalgamation Theorems in [4] are presented on a set-theoretical basis for pairs of plain graph rules without any application conditions. The Complement Rule and Multi-Amalgamation Theorems in this paper are valid in adhesive and weak adhesive HLR categories for n rules with application conditions [15]. These generalizations are non-trivial and important for applications of parallel graph transformations to communication-based systems [7], to model transformations from BPMN to BPEL [17], and for the modeling of the operational semantics of visual languages [8], where interaction schemes are used to generate multi-amalgamated rules and transformations based on suitable maximal matchings.

The theory of multi-amalgamation is a solid mathematical basis to analyze interesting properties of the operational semantics, like termination, local confluence, and functional behavior. However, it is left open for future work to

generalize the corresponding results in [2], like the Local Church–Rosser, Parallelism, and Local Confluence Theorems, to the case of multi-amalgamated rules, especially to the operational semantics of statecharts based on amalgamated graph transformation with maximal matchings in [13].

References

[1] Lack, S., Sobociński, P.: Adhesive Categories. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 273–288. Springer, Heidelberg (2004)

[2] Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs. Springer, Heidelberg (2006)

[3] Rozenberg, G. (ed.): Handbook of Graph Grammars and Computing by Graph Transformation. Foundations, vol. 1. World Scientific, Singapore (1997)

[4] Böhm, P., Fonio, H.R., Habel, A.: Amalgamation of Graph Transformations: A Synchronization Mechanism. JCSS 34(2-3), 377–408 (1987)

[5] Ehrig, H., Kreowski, H.J.: Parallelism of Manipulations in Multidimensional Information Structures. In: Mazurkiewicz, A. (ed.) MFCS 1976. LNCS, vol. 45, pp. 285–293. Springer, Heidelberg (1976)

[6] Taentzer, G., Beyer, M.: Amalgamated Graph Transformations and Their Use for Specifying AGG - an Algebraic Graph Grammar System. In: Ehrig, H., Schneider, H.-J. (eds.) Dagstuhl Seminar 1993. LNCS, vol. 776, pp. 380–394. Springer, Heidelberg (1994)

[7] Taentzer, G.: Parallel and Distributed Graph Transformation: Formal Description and Application to Communication Based Systems. PhD thesis, TU Berlin (1996)

[8] Ermel, C.: Simulation and Animation of Visual Languages based on Typed Algebraic Graph Transformation. PhD thesis, TU Berlin (2006)

[9] Löwe, M.: Algebraic Approach to Single-Pushout Graph Transformation. TCS 109, 181–224 (1993)

[10] Grønmo, R., Krogdahl, S., Møller-Pedersen, B.: A Collection Operator for Graph Transformation. In: Paige, R.F. (ed.) ICMT 2009. LNCS, vol. 5563, pp. 67–82. Springer, Heidelberg (2009)

[11] Hoffmann, B., Janssens, D., Eetvelde, N.: Cloning and Expanding Graph Transformation Rules for Refactoring. ENTCS 152, 53–67 (2006)

[12] Rensink, A., Kuperus, J.: Repotting the Geraniums: On Nested Graph Transformation Rules. ECEASST 18, 1–15 (2009)

[13] Golas, U., Biermann, E., Ehrig, H., Ermel, C.: A Visual Interpreter Semantics for Statecharts Based on Amalgamated Graph Transformation (Submitted 2010), `http://tfs.cs.tu-berlin.de/publikationen/Papers10/GBEE10.pdf`

[14] Golas, U.: Multi-Amalgamation in M-Adhesive Categories: Long Version. Technical Report 2010/05, Technische Universität Berlin (2010)

[15] Habel, A., Pennemann, K.H.: Correctness of High-Level Transformation Systems Relative to Nested Conditions. MSCS 19(2), 245–296 (2009)

[16] Ehrig, H., Habel, A., Lambers, L.: Parallelism and Concurrency Theorems for Rules with Nested Application Conditions. ECEASST 26, 1–23 (2010)

[17] Biermann, E., Ehrig, H., Ermel, C., Golas, U., Taentzer, G.: Parallel Independence of Amalgamated Graph Transformations Applied to Model Transformation. In: Engels, G., Lewerentz, C., Schäfer, W., Schürr, A., Westfechtel, B. (eds.) Essays Dedicated to M. Nagl. LNCS, vol. 5765. Springer, Heidelberg (2010)