Multi-Amalgamation of Rules with Application Conditions in $M$-Adhesive Categories

ULRIKE GOLAS$^1$, ANNEGRET HABEL$^2$ and HARTMUT EHRIG$^3$

$^1$ Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany, golas@zib.de
$^2$ Universität Oldenburg, Germany, annegret.habel@informatik.uni-oldenburg.de
$^3$ Technische Universität Berlin, Germany, ehrig@cs.tu-berlin.de

Received September 2011

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 nested 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 for $M$-adhesive categories, a slightly more general framework than (weak) adhesive HLR categories, for a bundle of rules with (nested) application conditions. The two main results are the Complement Rule Theorem showing how to construct a minimal complement rule for each subrule and the Multi-Amalgamation Theorem, which generalizes the well-known Parallelism and Amalgamation Theorems to the case of multiple synchronized parallelism. For the application of the largest amalgamated rule we use maximal matchings, which are computed depending on the actual instance graph. The constructions are illustrated by a small but meaningful running example, while a more complex case study concerning the firing semantics of Petri nets is presented as introductory example and motivation.

Contents

1 Introduction and Related Work 2
2 Firing Semantics of Petri Nets Using Amalgamation 5
3 Review of Basic Notions 8
4 Decomposition of Direct Transformations 11
5 Multi-Amalgamation 15
6 Multi-Amalgamation with Maximal Matchings 20
7 Conclusion 23
Appendix A Proofs of Facts and Theorems 26
Appendix B Additional Lemmas 33
1. Introduction and Related Work

1.1. Historical Background of Amalgamation

The concepts of adhesive (Lack and Sobociński, 2005) and weak adhesive high-level replacement (HLR) (Ehrig et al., 2006) categories have been a break-through for the double-pushout approach of algebraic graph transformations (Rozenberg, 1997). Almost all main results for graph transformation systems 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 (Ehrig et al., 2006). 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 (Ehrig et al., 2010b, 2012). In (Ehrig et al., 2010a) it is shown that also $\mathcal{M}$-adhesive categories, a slightly weaker version, are sufficient to formulate graph transformations in such a general categorical setting.

While most graph transformation models for distributed systems concentrate on the topological aspects of the system (Castellani and Montanari, 1983; Degano and Montanari, 1987), also the application of the main theorems for the analysis of such systems is of interest. One example is the Parallelism Theorem (Ehrig and Kreowski, 1976) stating that two parallel independent transformations can be combined and are equivalent to one transformation using the corresponding parallel rule. But for distributed systems, often a weaker form of parallel independence is required: two transformations do not have to be completely parallel independent, but may overlap on certain, well-defined elements dependently. This generalization of the Parallelism Theorem is called the Amalgamation Theorem, where the assumption of parallel independence is dropped and some synchronization takes place. It has been developed in (Böhm et al., 1987) on a set-theoretical basis for a pair of standard graph rules without application conditions.

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 so-called kernel morphism, which is a rule morphism from $p_0$ to $p_i$. Given two such kernel morphisms, the rules $p_1$ and $p_2$ can be glued along $p_0$ leading to an amalgamated rule $\tilde{p}$ which represents the synchronized effects of $p_1$ and $p_2$. Now two transformations via $p_1$ and $p_2$ are amalgamatable if they are parallel independent except for the elements matched by the kernel rule. In this case, similar to the Parallelism Theorem, the two transformations can be combined and are equivalent to one transformation using the amalgamated rule. This is the main statement of the Amalgamation Theorem: each amalgamable pair of transformations $G \Rightarrow G_i$ ($i = 1, 2$) via $p_1$ and $p_2$ leads to an amalgamated transformation $G \Rightarrow H$ via $\tilde{p}$.

Moreover, the Complement Rule Theorem in (Böhm et al., 1987) allows to construct a complement rule $\overline{p}$ out of a kernel morphism from $p_0$ to $p$. Using the kernel rule $p_0$ and the complement rule $\overline{p}$ we can construct a concurrent rule $p_0 \ast_E \overline{p}$ which is equal to $p$. Then the Concurrency Theorem allows to decompose each transformation $G \Rightarrow H$ via $p$ into sequences $G \Rightarrow G_i \Rightarrow H$ via $p_0$ and $\overline{p}$. Moreover, also an amalgamated transformation can be sequentialized this way.
1.2. Other Parallel Models of Computation in Graph Transformation

Parallel rewriting was first studied on the level of strings. Motivated by examples from biology, "Lindenmayer Systems", short L-systems, were developed as a mathematical theory of parallel languages in the 1970ies. The main idea of L-systems is to replace all letters of a string simultaneously according to a given set of rules. This idea was generalized to graphs leading to different kinds of parallel graph grammars and graph-L-systems (Rozenberg and Lindenmayer, 1976).

There are several other graph transformation-based approaches and tools which realize the transformation of multi-object structures. PROGRES (Schürr et al., 1999) and Fujaba (Fischer et al., 2000) feature so-called set-valued nodes which can be duplicated as often as necessary. Both approaches handle multi-objects in a pragmatic way. Object nodes are indicated to be matched optionally once, arbitrarily often, or at least once and adjacent arcs are treated accordingly. This concept focuses on multiple instances of single nodes instead of graph parts.

Further approaches that realize amalgamated graph transformation are AToM3, GReAT, and GROOVE. AToM3 supports the explicit definition of interaction schemes in different rule editors (de Lara et al., 2004) whereas GROOVE implements rule amalgamation based on nested graph predicates (Rensink and Kuperus, 2009). While nesting extends the expressiveness of these transformations, it is quite complicated to write and understand these predicates and it seems to be difficult to relate or integrate them to the theoretical results for graph transformation. The GReAT tool can use a group operator to apply delete, move, or copy operations to each match of a rule (Balasubramanian et al., 2007).

A related conceptual approach aiming at transforming collections of similar subgraphs is presented in (Grønmo et al., 2009). There, all collection operators (multi-objects) in a rule are replaced by the mapped number of collection match copies. Similarly, in (Hoffmann et al., 2006) a cloning operator is defined, where cloned nodes roughly correspond to multi-objects. But none of the aforementioned approaches investigates the formal analysis of amalgamated graph transformation.

1.3. Applications of Amalgamation

The concepts of amalgamation have been applied to communication based systems in (Taentzer and Beyer, 1994; Taentzer, 1996; Ermel, 2006) and transferred to the single-pushout approach of graph transformation in (Löwe, 1993). In (Biermann et al., 2010a), amalgamation is used to define a model transformation translating simple business process models written in the Business Process Modeling Notation (BPMN) to executable processes formulated in the Business Process Execution Language for Web Services (BPEL). It also plays a key role in the modeling of the operational semantics for visual languages (Ermel, 2006). A complex case study for the operational semantics of statecharts based on typed attributed graphs and multi-amalgamation is presented in (Golas et al., 2011; Golas, 2011). With amalgamation, we do not need helper structures or a complex external control structure to cover complex semantical steps in our approach. The result is a model-independent definition, which is not only visual and
intuitive but also allows to show termination and forms a solid basis for applying further
graph transformation-based analysis techniques.

The theory of amalgamation presented in this paper has been implemented in AGG
(Taentzer, 2004) and in our EMF transformation tool EMF Henshin (Biermann et al.,
2010b), which has been extended by visual editors for amalgamated rules and application
conditions [BESW10].

1.4. The Aim of this Paper

In most of the 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. While some of the tools provide
an ad-hoc implementation of multi-amalgamation, the underlying theory is not elabo-
rated. 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 $\mathcal{M}$-adhesive
systems based on rules with nested application conditions. In Ehrig et al. (2012), the
amalgamation of exactly two rules is shortly described in this framework. Our work in
this paper 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 (Ehrig et al., 2006).

The work in this paper extends the one in (Golas et al., 2010) in several ways: First,
we consider amalgamated transformations in any $\mathcal{M}$-adhesive category, while in (Golas
et al., 2010) only adhesive categories where used. Second, we present as a new case study
the firing semantics of Petri nets. While this semantics is much smaller and easier to
survey than the one of statecharts in (Golas et al., 2011), it still shows the importance of
multi-amalgamation including the use of application conditions. Moreover, we give the
full proofs for the results and extend the theory by maximal matchings which allow to
compute the maximal amalgamated rule applicable at a certain kernel match.

1.5. Organization of this Paper

This paper is organized as follows. In Section 2, we discuss how to define the semantics of
Petri nets using graph transformation and show, that amalgamation eases the rule defi-
nition without the need for additional control structure. In Section 3, we review basic no-
tions of $\mathcal{M}$-adhesive categories, transformations, and application conditions. In Section 4,
we introduce kernel rules, multi rules, and kernel morphisms leading to the Complement
Rule Theorem as first main result. In Section 5, we construct multi-amalgamated rules
and transformations and show as second main result the Multi-Amalgamation Theorem.
Maximal matchings, which are used to compute the maximal amalgamated rule, are con-
structed in Section 6. In Section 7, we present a summary of our results and discuss
future work. All more complex proofs can be found in Appendix A, while in Appendix B
some technical lemmas underlying these proofs are shown.
2. Firing Semantics of Petri Nets Using Amalgamation

A Petri net, or place/transition net (Reisig and Rozenberg, 1998), consists of places (circles) and transitions (rectangles) with arcs between them. A place with a connecting arc to or from a transition is called its pre or post place, respectively. Note that for simplicity we forbid a place to be both a pre and a post arc of the same transition. A number of tokens is put on each place, where an arbitrary number of tokens is allowed. Natural numbers at the arcs mark how many tokens are moved when the transition fires. Note, that no number at an arc abbreviates 1. A transition is enabled if all its pre places hold at least as many tokens as required by the arc inscription. Firing this transition leads to the deletion of this number of tokens on the pre places and the respective number of tokens is added to each post place (see Fig. 1). For the modeling of the nets, we use typed attributed graphs (Ehrig et al., 2006), which we do not introduce here in detail. For each place, there is an attribute token of type integer representing the number of tokens at this place. In the figures, we simply depict this number inside the place.

In general, for the definition of a rule-based semantics of models two main approaches are known in the literature: First, the rules can be dependent on the actual instance of the model (Kuske et al., 2002), thus we have some rule schemes or instructions which have to be applied describing how to obtain the semantical rule for a concrete semantical step dependent on how the model instance looks like. In a place/transition net, for a transition with $m$ pre and $n$ post places we have variables $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ denoting the number of tokens for the rule.

Given the arc weights $a_1, \ldots, a_m$ and $b_1, \ldots, b_n$ for the pre and post arcs, this leads to a rule \texttt{fire}_a_{1, \ldots, a_m, b_1, \ldots, b_n} (see Fig. 2) describing the token handling. We need application conditions to make sure that all pre and post places are matched. In addition, we have to check that the number of tokens at a pre place is not smaller then the corresponding arc weight. This rule scheme can be interpreted for each occurring transition thus defining the semantics of a concrete place/transition net. Note that to obtain all firing rules of

---

**Fig. 1.** The firing of the transition $t$

**Fig. 2.** The rule scheme for firing an arbitrary transition in place/transition nets
+ application conditions:
all pre-places: $a_i \leq x_i$
no place marked
no other transition marked

+ application condition:
place
not marked

+ application conditions:
all pre- and
post-places marked

+ application conditions:
no transition marked

Fig. 3. The general rules for firing a transition in place/transition nets

place/transition nets, we have to consider all combinations of values for $m$, $n$, $a_i$, and $b_j$. This approach is easy to use once the rules are constructed, but when changing a model also the semantical rules have to be adapted. For arbitrary instances not known in advance, infinitely many rules appear, which are difficult to analyze.

For the second approach, general rules are applied according to some complex control structure (Varró, 2002). For place/transition nets, first we have to mark an active transition to declare its firing (top rule in Fig. 3). Since we do not know in advance how many pre and post places have to be handled, we need one rule deleting a token from one pre place, and one rule for adding a token to one post place of a transition (middle rules in Fig. 3). Since we have to know which places were already processed we also have to mark these places. In the end, when all of the involved places are handled, the transition and afterwards the places can be unmarked (bottom rules in Fig. 3). Applying the first rule $m$-times and the second one $n$-times with the corresponding matches leads to a firing step in the Petri net. Thus, all model instances are handled using the same rules. But even for this simple example, a lot of marking is needed to ensure the right matches. Although the single rules are relatively easy to understand, the additional helper structure, often combined with complex control structure for more difficult examples, makes it hard to understand the complete semantics.

Even for Petri nets, whose semantics can be described relatively easy on a set-theoretical basis (Reisig and Rozenberg, 1998), both graph transformation approaches discussed above have their drawbacks. When we analyze the second approach, it is obvious that the marking of the transition represents a kind of synchronization: instead of arbitrary matches for the transition, the handling of the pre and post places has to happen at this marked transition. The marking of the pre and post places is necessary to avoid multiple processing of the same place. Both markings are not necessary for the first approach, because there all places are handled at the same time. Our goal is to combine
both approaches leading to a universal rule application for all model instances with less additional structure such that analysis becomes easier. With amalgamation, we define an interaction scheme which provides the necessary rules. Using maximal matchings the semantical step for each model instance can be computed. As shown in the following, for place/transition nets we only need one kernel rule and two multi rules to describe the complete firing semantics for all well-defined nets. We neither need infinitely many rules, which are difficult to analyze, nor any control or helper structure when using amalgamation. This eases the modeling of the semantics and prevents errors.

The semantics for place/transition nets using amalgamation is shown in Fig. 4. The kernel rule $p_0$ is depicted twice in the top row. Note that we use rules in the double-pushout approach with a left-hand side $L$ describing what has to be find to apply the rule, an interface $K$ describing what is preserved, and a right-hand side $R$ showing the resulting graph part. This means that the elements $L \setminus K$ are deleted, while the elements $R \setminus K$ are created by the rule. The kernel rule selects an activated transition (but does not change or mark it) and controls the synchronization. Note that we use an application condition $ac_0$, shown in the middle of Fig. 4, saying that for all morphisms $a_0$ the attribute value of the arc $a$ has to be smaller than that of the node $x$. We have two multi rules which define the handling of the tokens: $p_1$ on the left for the pre places selecting a place and decreasing the number of tokens, and $p_2$ on the right for the post places selecting a place and increasing the number of tokens. We define morphisms $s_1$ and $s_2$ from the kernel rule to the multi rules which form an interaction scheme. Whenever a firing step is performed, we compute a maximal weakly disjoint matching meaning that we look for matches for the multi rules that overlap on the kernel rule, but are disjoint outside. Such a matching is relatively easy and inexpensively to compute and ensures that all pre and post places of a chosen transition are mapped.

For example, the maximal weakly disjoint matching for the firing of the transition $t$ in Fig. 1 with kernel match $t_0$ includes three matches for the multi rule $p_1$ and two matches for the multi rule $p_2$, one for each pre and post place, respectively. Amalgamation of this maximal weakly disjoint matching leads to the amalgamated rule $\tilde{p}_s$ shown in Fig. 5.
3. Review of Basic Notions

The basic idea of adhesive categories (Lack and Sobociński, 2005) is to have a category with pushouts and pullbacks along monomorphisms 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 (Lack and Sobociński, 2005; Ehrig et al., 2006)), including the standard category of graphs which is used as a running example in this paper.

\( M \)-adhesive categories include a distinguished morphism class \( M \) of monomorphisms and extend adhesive categories with suitable properties. As a main difference, they only require pushouts along \( M \)-morphisms to be vertical weak van Kampen squares.

**Definition 3.1 (Van Kampen square).**
A pushout as at the bottom of the cube on the right with \( m \in M \) is a vertical weak van Kampen square, short \( M \)-van Kampen square, if it satisfies the vertical weak van Kampen property, i.e., for any commutative cube, where the back faces are pullbacks and the vertical morphisms \( b, c, d \in M \), the following statement holds: The top face is a pushout if and only if the front faces are pullbacks.

In contrast, the horizontal weak van Kampen property assumes that \( f \in M \) instead of \( b, c, d \in M \), while the (standard) van Kampen property does not require any additional \( M \)-morphisms.

**Definition 3.2 (\( M \)-adhesive category).** An \( M \)-adhesive category \((C, M)\) consists of a category \( C \) and a class \( M \) of monomorphisms in \( C \), which is closed under isomorphisms, composition, and decomposition (\( g \circ f \in M \) and \( g \in M \) implies \( f \in M \)), such that \( C \) has pushouts and pullbacks along \( M \)-morphisms, \( M \)-morphisms are closed under pushouts and pullbacks, and pushouts along \( M \)-morphisms are \( M \)-van Kampen squares.
Well-known examples of $\mathcal{M}$-adhesive categories are the categories $(\text{Sets}, \mathcal{M})$ of sets, $(\text{Graphs}, \mathcal{M})$ of graphs, $(\text{Graphs}_{\text{TG}}, \mathcal{M})$ of typed graphs, $(\text{ElemNets}, \mathcal{M})$ of elementary Petri nets, $(\text{PTNets}, \mathcal{M})$ of place/transition nets, where for all these categories $\mathcal{M}$ is the class of all monomorphisms, and $(\text{AGraphs}_{\text{ATG}}, \mathcal{M})$ of typed attributed graphs, where $\mathcal{M}$ is the class of all injective typed attributed graph morphisms with isomorphic data type component (see (Ehrig et al., 2006)).

In the double-pushout approach to transformations, rules describe in a general way how to transform objects. The application of a rule to an object is called a transformation and based on two gluing constructions, which are pushouts in the corresponding category.

**Definition 3.3 (Rule and 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 $\mathcal{M}$-morphisms $l$ and $r$. An application of such a rule to an object $G$ via a match $m : L \rightarrow G$ is constructed as two pushouts (1) and (2) leading to a direct transformation $G \xrightarrow{\tilde{p},\tilde{m}} H$.

**Example 3.1.** An example for a rule can be found in Fig. 10 in the top row. The application of the rule to the graph $G$ leads to the depicted transformation $G \xrightarrow{\tilde{p},\tilde{m}} H$, where both squares are pushouts.

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 $a : L \rightarrow C$, demanding a certain structure in addition to $L$, and also negative application conditions $\neg \exists a$, forbidding such a structure. A match $m : L \rightarrow G$ satisfies $\exists a$ (resp. $\neg \exists a$) if there is a (resp. no) $\mathcal{M}$-morphism $q : C \rightarrow G$ satisfying $q \circ a = m$. In more detail, we use nested application conditions (Habel and Pennemann, 2009), short application conditions.

**Definition 3.4 (Application condition and satisfaction).** An application condition $ac$ over an object $L$ is of the form $ac = true$ or $ac = \exists(a, ac')$, where $a : L \rightarrow C$ is a morphism and $ac'$ is an application condition over $C$. Given a condition $ac$ over $L$, then a morphism $m : L \rightarrow G$ satisfies $ac$, written $m \models ac$, if $ac = true$ or $ac = \exists(a, ac')$ and there exists a morphism $q \in \mathcal{M}$ with $q \circ a = m$ and $q \models ac'$.

Moreover, application conditions are closed under Boolean formulas (with finite or infinite index set) and satisfaction is extended as usual. For simplification, false abbreviates $\neg true$, $\exists a$ abbreviates $\exists(a, true)$, and $\forall(a, ac)$ abbreviates $\neg \exists(a, \neg ac)$. With $ac_C \equiv ac'_C$, we denote the semantical equivalence of $ac_C$ and $ac'_C$ on $C$.

**Example 3.2.** In Fig. 10, the application condition $\tilde{ac}_s$ of the rule $\tilde{p}_s$ is stated above the rule, while the involved morphisms are shown on the right. It forbids various edges from or to node 1. The match morphism $\tilde{m}$ satisfies this application condition.

In this paper we consider rules of the form $p = (L \xleftarrow{l} K \xrightarrow{r} R, ac)$, 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 (Habel and Pennemann, 2009; Ehrig et al., 2010b).

For the shift construction over morphisms we use a distinguished class \( \mathcal{E}' \) of morphism pairs with the same codomain such that for any pair of morphisms with common codomain a unique \( \mathcal{E}'-\mathcal{M} \) pair factorization exists.

**Definition 3.5 (Shift over morphism).** Given an application condition \( ac = \exists(a, ac') \) over \( P \) and a morphism \( b : P \to P' \), then \( \text{Shift}(b, ac) \) is an application condition over \( P' \) defined by \( \text{Shift}(b, ac) = \exists(a', b') \in F \exists(a', \text{Shift}(b', ac')) \) with \( F = \{(a', b') \mid (a', b') \in \mathcal{E}', b' \in \mathcal{M}, b' \circ a = a' \circ b \} \). Moreover, \( \text{Shift}(b, \text{true}) = \text{true} \) and the construction is extended for Boolean formulas in the usual way.

**Remark 3.1.** \( F \) is finite if \( \mathcal{E}' \) consists of jointly surjective pairs of morphisms, which is the case in our example categories.

**Example 3.3.** An example for shifting an application condition over a morphism can be found in the left of Fig. 7. We have that \( \text{Shift}(v_1, \neg \exists a_1') = \neg \exists a_1' \), because square \( (\ast) \) is the only possible commuting square leading to \( a_1', b_1' \) jointly surjective and \( b_1' \) injective.

**Fact 3.1.** Given an application condition \( ac \) over \( P \) and morphisms \( b : P \to P' \) and \( p : P' \to G \), then \( p \models \text{Shift}(b, ac) \) if and only if \( p \circ b \models ac \).

**Proof.** See (Habel and Pennemann, 2009; Ehrig et al., 2010b).

In analogy to the application condition over \( L \), which is a pre application condition, it is also possible to define post application conditions over the right hand side \( R \) of a rule. Since these application conditions over \( R \) can be translated to equivalent application conditions over \( L \) (and vice versa) (Habel and Pennemann, 2009), we can restrict our rules to application conditions over \( L \).

**Definition 3.6 (Shift over rule).** Given a rule \( p = (L \xleftarrow{i} K \xrightarrow{r} R, ac) \) and an application condition \( ac_R = \exists(a, ac'_R) \) over \( R \), then \( L(p, ac_R) \) is an application condition over \( L \) defined by \( L(p, ac_R) = \exists(b, L(p^*, ac'_R)) \) if \( a \circ r \) has a pushout complement (1) and \( p^* = (Y \xleftarrow{i} Z \xrightarrow{r} X) \) is the derived rule by constructing pushout (2), otherwise false. Moreover, \( L(p, \text{true}) = \text{true} \) and the construction is extended to Boolean formulas in the usual way.
Example 3.4. An example for shifting an application condition over a rule shown by the two pushout squares \((PO_1)\) and \((PO_2)\) in Fig. 7, where \(L(p_1^*, \neg \exists a_{11}) = \neg \exists a_1\).

Fact 3.2. Given a transformation \(G \xrightarrow{p,m} H\) via a rule \(p = (L \xleftarrow{l} K \xrightarrow{r} R, ac)\) and an application condition \(ac_R\) over \(R\), then we have that \(m \models L(p, ac_R)\) if and only if \(n \models ac_R\).

Proof. See (Habel and Pennemann, 2009).

Shifts over morphisms are compositional and shifts over morphisms and rules are compatible via double pushouts.

Fact 3.3. Given an application condition \(ac\) on \(R\), the double pushouts (1) and (2) and morphisms \(a, b\), then we have that

\[
\text{— Shift}(b, \text{Shift}(a, ac)) \cong \text{Shift}(b \circ a, ac),
\]

\[
\text{— Shift}(m, L(p, ac)) \cong L(p', \text{Shift}(n, ac)).
\]

Proof. See (Habel and Pennemann, 2009; Ehrig et al., 2010b).

General Assumptions

In this paper we assume to have an \(\mathcal{M}\)-adhesive category with binary coproducts, initial pushouts, \(\mathcal{E}'\)-\(\mathcal{M}\)-pair factorization, and effective pushouts (Ehrig et al., 2006; Golas, 2011). We consider rules with (nested) application conditions (Habel and Pennemann, 2009) as explained above and assume that the reader is familiar with parallelism and concurrency in the sense of (Ehrig et al., 2006). Moreover, we use the corresponding constructions and results with application conditions in (Ehrig et al., 2010b). In the following, a bundle represents a family of morphisms or transformation steps with the same domain, which means that a bundle always starts at the same object.

4. Decomposition of Direct Transformations

In this section, we show how to decompose a direct transformation in \(\mathcal{M}\)-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 5. 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 4.1 (Kernel morphism). Given rules \( p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, ac_0) \) and \( p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, ac_1) \), a kernel morphism \( s_1 : p_0 \to p_1 \), \( s_1 = (s_{1,L}, s_{1,K}, s_{1,R}) \) consists of \( \mathcal{M} \)-morphisms \( s_{1,L} : L_0 \to L_1 \), \( s_{1,K} : K_0 \to K_1 \), and \( s_{1,R} : R_0 \to R_1 \) such that in the following diagram (1) and (2) are pullbacks, (1) has a pushout complement (1') for \( s_{1,L} \circ l_0 \), and \( ac_0 \) and \( ac_1 \) are complement-compatible w.r.t. \( s_1 \), i.e. given pushout (3) then \( ac_1 \cong \text{Shift}(s_{1,L}, ac_0) \land \text{L}(p_1, \text{Shift}(v_1, ac_1')) \) for some \( ac_1' \) on \( L_{10} \) and \( p_1^* = (L_1 \xleftarrow{l_1} L_{10} \xrightarrow{v_1} E_1) \). In this case, \( p_0 \) is called kernel rule and \( p_1 \) multi rule.

Remark 4.1. 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.

Example 4.1. 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

Fig. 6. The kernel rule \( p_0 \) and the multi rules \( p_1 \) and \( p_2 \)
Multi-Amalgamation in \( \mathcal{M} \)-Adhesive Categories

Fig. 7. Constructions for the application conditions

depicted in the top of Fig. 6. 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.

In the middle and bottom of Fig. 6, 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 \( s_1 : p_0 \rightarrow p_1 \) as shown in the top of Fig. 6 with pullbacks \((1_1^1)\) and \((2_1^1)\), and pushout complement \((1_1)'\). For the application conditions, \( ac_1 = \text{Shift}(s_1, L, ac_0) \land \neg \exists a_1 \equiv \text{Shift}(s_1, L, ac_0) \land L(p_1^*, \text{Shift}(v_1, \neg \exists a_1')) \) as shown in the left of Fig. 7. Thus \( ac_1' = \neg \exists a_1' \), and \( ac_0 \) and \( ac_1 \) are complement compatible.

Similarly, there is a kernel morphism \( s_2 : p_0 \rightarrow p_2 \) as shown in the bottom of Fig. 6 with pullbacks \((1_2^2)\) and \((2_2^2)\), 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 4.1 (Existence of complement rule).** Given rules \( p_0 = (L_0 \leftarrow K_0 \rightarrow R_0, ac_0) \) and \( p_1 = (L_1 \leftarrow K_1 \rightarrow R_1, ac_1) \), and a kernel morphism \( s_1 : p_0 \rightarrow p_1 \) then there is a canonical way to construct a rule \( \overline{p} = (L_1 \leftarrow \overline{K}_1 \rightarrow R_1, \overline{ac}) \) and a jointly epimorphic cospan \( R_0 \leftarrow E_1 \rightarrow \overline{L}_1 \) such that the \( E_1 \)-concurrent rule \( p_0 *_{E_1} \overline{p} \) exists.
and \( p_1 = p_0 \ast_{E_1} \overline{P}_1 \). For the definition of \( E \)-concurrent rules for rules with application conditions see (Ehrig et al., 2010b).

**Proof.** See Subsection A.1 in the appendix.

**Remark 4.2.** Note, that using the construction in the appendix the interface \( K_0 \) of the kernel rule has to be preserved in the complement rule. This canonical construction of \( \overline{P}_1 \) is not unique w.r.t. the property \( p_1 = p_0 \ast_{E_1} \overline{P}_1 \), since other choices for \( S_1 \) with \( M \)-morphisms \( s_{11} \) and \( s_{13} \) also lead to a well-defined construction. In particular, one could choose \( S_1 = R_0 \) leading to \( \overline{P}_1 = E_1 \leftarrow R_{10} \rightarrow R_1 \). Our choice represents a smallest possible complement, which should be preferred in most application areas.

**Definition 4.2 (Complement rule).** Given rules \( p_0 = (L_0 \leftarrow K_0 \rightarrow R_0, ac_0) \) and \( p_1 = (L_1 \leftarrow K_1 \rightarrow R_1, ac_1) \), and a kernel morphism \( s_1 : p_0 \rightarrow p_1 \) then the canonical rule \( p_1 = (L_1 \leftarrow K_1 \rightarrow R_1, ac_1) \) identified by Thm. 4.1 is called complement rule (of \( s_1 \)).

**Example 4.2.** Consider the kernel morphism \( s_1 \) depicted in Fig. 6. Using the construction in Thm. 4.1 we obtain the complement rule in the top row in Fig. 8 with the application condition \( ac_1 = \neg \exists a_1 \) constructed in the right of Fig. 7. In Fig. 9, the diagrams of the construction are shown. Similarly, we obtain a complement rule for the kernel morphism \( s_2 : p_0 \rightarrow p_2 \) in Fig. 6, which is depicted in the bottom row of Fig. 8.

![Fig. 8. The complement rules for the kernel morphisms](image)

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 4.1 (Decomposition of transformations).** Given rules \( p_0 = (L_0 \leftarrow K_0 \rightarrow_{R_0, ac_0}) \) and \( p_1 = (L_1 \leftarrow K_1 \rightarrow_{R_1, ac_1}), \) a kernel morphism \( s_1 : p_0 \rightarrow p_1 \), and a direct transformation \( t_1 : G \xrightarrow{p_1,m_1} G_1 \) then \( t_1 \) can be decomposed into the transformation \( G \xrightarrow{p_0,m_0} G_0 \xrightarrow{\overline{P}_1} G_1 \) with \( m_0 = m_1 \circ s_{1,L} \) where \( \overline{P}_1 \) is the complement rule of \( s_1 \).
Multi-Amalgamation in $\mathcal{M}$-Adhesive Categories

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure9}
\caption{The construction of the complement rule for the kernel morphism $s_1$}
\end{figure}

\textbf{Proof.} We have that $p_1 \cong p_0 \ast E_1 \mu_1$. The analysis part of the Concurrency Theorem (Ehrig \textit{et al.}, 2010b) now implies the decomposition into $G \xrightarrow{p_0,m_0} G_0 \xrightarrow{p_1,m_1} G_1$ with $m_0 = m_1 \circ s_{1,L}$.

\section{5. Multi-Amalgamation}

In (Böhm \textit{et al.}, 1987), 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 (Ehrig and Kreowski, 1976), 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 $\mathcal{M}$-adhesive category.

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 5.1 (Amalgamated rule).**
Given rules $p_i = (L_i \leftarrow K_i \rightarrow R_i, ac_i)$ 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 amalgamated rule $\tilde{p}_s = (\tilde{L}_s \leftarrow \tilde{K}_s \rightarrow \tilde{R}_s, \tilde{ac}_s)$ is constructed as the componentwise colimit of the kernel morphisms:

- $\tilde{L}_s = \text{Col}((s_i, L)_{i=1,\ldots,n})$, $\tilde{K}_s = \text{Col}((s_i, K)_{i=1,\ldots,n})$, $\tilde{R}_s = \text{Col}((s_i, R)_{i=1,\ldots,n})$,
- $\tilde{l}_s$ and $\tilde{r}_s$ are induced by $(t_i, L \circ l_i)_{i=0,\ldots,n}$ and $(t_i, R \circ r_i)_{i=0,\ldots,n}$, respectively,
- $\tilde{ac}_s = \bigwedge_{i=1,\ldots,n} \text{Shift}(t_i, L, ac_i)$.

**Fact 5.1.** 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.** See Subsection A.2 in the appendix.

The application of an amalgamated rule yields an amalgamated transformation.

**Definition 5.2 (Amalgamated transformation).** The application of an amalgamated rule to a graph $G$ is called an amalgamated transformation.
Example 5.1. Consider the bundle $s = (s_1, s_2, s_3 = s_1)$ of the kernel morphisms depicted in Fig. 6. The corresponding amalgamated rule $\tilde{p}_s$ is shown in the top row of Fig. 10. This amalgamated rule can be applied to the graph $G$ leading to the amalgamated transformation depicted in Fig. 10, 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 5.3 ($s$-amalgamable). Given a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,...,n}$, a bundle of direct transformations steps $(G \xrightarrow{\nu, m_i} G_i)_{i=1,...,n}$ is $s$-amalgamable, if

- 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 $p_{ij} : L_{00} \to D_j$ and $p_{ji} : L_{j0} \to D_i$ such that $f_j \circ p_{ij} = m_i \circ u_i$, $f_i \circ p_{ji} = m_j \circ u_j$, $g_j \circ p_{ij} = ac'_i$, and $g_i \circ p_{ji} = ac'_j$.

![Diagram](https://example.com/diagram.png)

Similar to the characterization of parallel independence in (Ehrig et al., 2006) we can give a set-theoretical characterization of weak independence without application conditions.

Fact 5.2. For graphs and other set-based structures, weakly independent matches without considering the application conditions 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 $i \neq j$, 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 prove 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.
"c" 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 $y_i = u_i(z_i) \in u_i(L_0 \setminus w_i(K_0))$, and $x = m_i(u_i(z_i)) = f_j(p_j(z_i)) = m_j(y_j)$, and by pushout properties $y_j \in l_j(K_j)$ and $x \in m_j(l_j(K_j))$. Similarly, $x \in m_i(l_i(K_i))$.

"s" For $x \in L_0, x = w_i(k)$ define $p_{ij}(x) = k_j(s_j, K(k))$, then $f_j(p_{ij}(x)) = f_j(k_j(s_j, K(k))) = m_j(l_j(s_j, K(k))) = m_j(s_j, L(l_0(k))) = m_i(u_i(w_i(k))) = m_i(u_i(x))$. 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 5.2. Consider the bundle $s = (s_1, s_2, s_3 = s_1)$ of kernel morphisms from Ex. 5.1. For the graph $G$ given in Fig. 10 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. 11. 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 5.3. Given a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,...,n}$ and an s-amalgamable bundle of direct transformations $(G \xrightarrow{p_i, m_i} G_i)_{i=1,...,n}$ then each direct transformation $G \xrightarrow{p_i, m_i} G_i$ can be decomposed into a transformation $G \xrightarrow{p_0, m_0} G_0 \xrightarrow{p_i, m_i} G_i$. Moreover, the transformations $G_0 \xrightarrow{p_i, m_i} G_i$ are pairwise parallel independent.

Proof. See Subsection A.3 in the appendix.
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 5.1 (Multi-Amalgamation).** Consider a bundle of kernel morphisms $s = (s_i : p_0 \rightarrow p_i)_{i=1,...,n}$.

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

2 *Analysis.* Given an amalgamated transformation $G \xrightarrow{\tilde{p}_s,m_s} H$ then there are $s_i$-related transformations $G \xrightarrow{p_i,m_i} G_i \xrightarrow{\tilde{q}_i,m_i} H$ for $i = 1,\ldots,n$ such that $G \xrightarrow{p_i,m_i} G_i$ is $s$-amalgamable.

3 *Bijective Corresponcence.* The synthesis and analysis constructions are inverse to each other up to isomorphism.

**Proof.** See Subsection A.4 in the appendix.

**Remark 5.1.** Note, that $q_i$ can be constructed as the amalgamated rule of the kernel morphisms $(p_{K_0} \rightarrow \overline{p_j})_{j \neq i}$, where $p_{K_0} = (K_0 \xleftarrow{id_{K_0}} K_0 \xrightarrow{id_{K_0}} K_0, \text{true})$ 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 (Böhm et al., 1987). Moreover, if $p_0$ is the empty rule, this is the Parallelism Theorem in (Ehrig et al., 2010b), since the transformations are parallel independent for an empty kernel match.

**Example 5.3.** As already stated in Example 5.2, the transformations $G \xrightarrow{p_1,m_1} G_1$, $G \xrightarrow{p_2,m_2} G_2$, and $G \xrightarrow{p_1,m_2} G_3$ shown in Fig. 11 are $s$-amalgamable for the bundle $s = (s_1,s_2,s_3 = s_1)$ of kernel morphisms. Applying Fact 5.3, we can decompose these transformations into a transformation $G \xrightarrow{p_0,m_0} G_0$ followed by transformations

![Fig. 12. The decomposition of the s-amalgamable bundle](image-url)
G₀ = \[ \pi₁, m₁ \Rightarrow G₁ \], \( G₀ = \pi₂, m₂ \Rightarrow G₂ \), and \( G₀ = \pi₃, m₃ \Rightarrow G₃ \) via the complement rules, which are pairwise parallel independent. These transformations are depicted in Fig. 12. Moreover, Thm. 5.1 implies that we obtain for this bundle of direct transformations an amalgamated transformation \( G = \tilde{\pi}_s, \tilde{m} \Rightarrow H \), which is the transformation already shown in Fig. 10. Vice versa, the analysis of this amalgamated transformation leads to the s-amalgamable bundle of transformations \( G = \pi₁, m₁ \Rightarrow G₁ \), \( G = \pi₂, m₂ \Rightarrow G₂ \), and \( G = \pi₃, m₃ \Rightarrow G₃ \) from Fig. 11.

6. Multi-Amalgamation with Maximal Matchings

An important extension of the presented theory is the introduction of interaction schemes and maximal matchings. For many interesting application areas, including the operational semantics for Petri nets and statecharts, we do not want to define the matches for the multi rules explicitly, but to obtain them dependent on the object to be transformed. For example, for the firing semantics of statecharts (Golas et al., 2011), 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. Similarly, for our Petri net semantics introduced in Section 2, the pre and post places of a transition should be computed during runtime and dependent on the current Petri net model.

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 rule match. For maximal weakly independent matchings, we require the matchings of the multi rules to be weakly independent to ensure that the resulting bundle of transformations is amalgamable. This is the minimal requirement to meet the definition.

**Definition 6.1 (Interaction scheme).** A kernel rule \( p₀ \) and a set of multi rules \( \{p₁, \ldots, p_k\} \) with kernel morphisms \( s_j : p₀ \rightarrow p_j \) for \( j = 1, \ldots, k \) form an interaction scheme \( is = \{s₁, \ldots, s_k\} \).

When given an interaction scheme, we want to apply as many rules occurring in the interaction scheme as often as possible over a certain kernel rule match. For maximal weakly independent matchings, we require the matchings of the multi rules to be weakly independent to ensure that the resulting bundle of transformations is amalgamable. This is the minimal requirement to meet the definition.

**Definition 6.2 (Maximal weakly independent matching).** Given an interaction scheme \( is = \{s₁, \ldots, s_k\} \) with \( s_j : p₀ \rightarrow p_j \) for \( j = 1, \ldots, k \) and a family of matchings \( m = (m_i : L'_i \Rightarrow G) \), where each \( p'_i \) corresponds to some \( p_j \) for \( j \leq k \), with transformations \( G = \pi'_i, m_i \Rightarrow G_i \), then \( m \) forms a maximal weakly independent matching if the bundle \( G = \pi'_i, m_i \Rightarrow G_i \) is multi-amalgamable and for any rule \( p_j \) no other match \( m' : L_j \Rightarrow G \) can be found such that \(((m_i), m')\) fulfills this property.

This definition directly leads to the following algorithm to compute maximal weakly independent matchings for graphs and graph-like structures.

**Algorithm 6.1 (Maximal weakly independent matching).** Given a graph \( G \) and an interaction scheme \( is = \{s₁, \ldots, s_k\} \), a maximal weakly disjoint matching \( m = (m₀, m₁, \ldots, m_n) \) can be computed as follows:
1. Set $i = 0$. Choose a kernel matching $m_0 : L_0 \to G$ such that $G \xrightarrow{p_0,m_0} G_0$ is a valid transformation.

2. As long as possible: Increase $i$, choose a multi rule $\hat{p}_i = p_j$ with $j \in \{1, \ldots, k\}$, and find a match $m_i : L_j \to G$ such that $m_i \circ s_{j,L} = m_0$, $G \xrightarrow{p_j,m_i} G_i$ is a valid transformation, the matches $m_1, \ldots, m_i$ are weakly independent, and $m_i \neq m_\ell$ for all $\ell = 1, \ldots, i - 1$.

3. If no more valid matches for any rule in the interaction scheme can be found, return $m = (m_0, m_1, \ldots, m_n)$.

The maximal weakly independent matching leads to a bundle of kernel morphisms $s = (s_1 : p_0 \to \hat{p}_i)$ and an $s$-amalgamable bundle of direct transformations $G \xrightarrow{\hat{p}_i,m_i} G_i$.

For applications, the computation of maximal weakly independent matchings needs a lot of backtracking, because often a match in Step 2 is not weakly independent from an already chosen one, which has to be checked pairwise for this new match compared to all others. While the application conditions always have to be analyzed, since they may state global properties of the resulting graph, at least for the elements available for the new match some restrictions may help to enhance the computation. In many cases, it is adequate to require the matches to be disjoint outside the kernel match. A typical example is the semantics of Petri nets as described in Section 2 - there, all maximal weakly independent matchings are also weakly disjoint. This disjointness property is described formally by a certain pullback requirement. Using maximal weakly disjoint matchings for implementation, we can rule out model parts that were already matched.

**Definition 6.3 (Maximal weakly disjoint matching).**

Given an interaction scheme $is = \{s_1, \ldots, s_k\}$ and a maximal weakly independent matching $m = (m_i : L'_i \to G)$ then $m$ forms a **maximal weakly disjoint matching** if the square $(P_{\ell})$ is a pullback for all $i \neq \ell$.

Note that for maximal weakly disjoint matchings, the pullback requirement already implies the existence of the morphisms for the weakly independent matches. Only the property for the application conditions has to be checked in addition.

**Fact 6.1.** Given an object $G$, a bundle of kernel morphisms $s = (s_1, \ldots, s_n)$, and matches $m_1, \ldots, m_n$ leading to a bundle of direct transformations $G \xrightarrow{\hat{p}_i,m_i} G_i$ such that $m_i \circ s_{i,L} = m_0$ and square $(P_{\ell})$ is a pullback for all $i \neq \ell$ then the bundle $G \xrightarrow{\hat{p}_i,m_i} G_i$ is $s$-amalgamable for transformations without application conditions.

**Proof.** By construction, the matches $m_i$ agree on the match $m_0$ of the kernel rule. It remains to show that they are weakly independent.

Consider the transformations $G \xrightarrow{\hat{p}_i,m_i} G_i$ with pushouts (20) and (21) in the following diagram. For the cube on the right, the bottom face is pushout (20), the back right face is pullback (11), and the front right face is pullback $(P_{\ell})$. Now construct the pullback of $f_i$ and $m_\ell$ as the front left face, and from $m_\ell \circ s_{i,L} \circ l_0 = m_i \circ s_{i,L} \circ l_0 = m_i \circ l_0 \circ s_{i,K} = f_i \circ k_0 \circ s_{i,K}$ we obtain a morphism $p$ with $f \circ p = s_{i,L} \circ l_0$ and $m \circ p = k_0 \circ s_{i,K}$.
From pullback composition and decomposition of the right and left faces it follows that also the back left face is a pullback. Now the $\mathcal{M}$-van Kampen property can be applied leading to a pushout in the top face. Since pushout complements are unique up to isomorphism, we can substitute the top face by pushout $\left(1'\right)$ from Def. 5.3 with $P \cong L_0$. Thus we have found the morphism $p_{i\ell} := \hat{m}$ with $f_i \circ p_{i\ell} = m_{i\ell} \circ u_i$. This construction can be applied for all pairs $i, \ell$ leading to weakly independent matches without application conditions.

This fact leads to a set-theoretical characterization of maximal weakly disjoint matchings similar to the result in Fact 5.2.

**Fact 6.2.** For graphs and graph-based structures, valid matches $m_0, m_1, \ldots, m_n$ with $m_i \circ s_{i,L} = m_0$ for all $i = 1, \ldots, n$ form a maximal weakly disjoint matching without application conditions if and only if $m_i(L_i) \cap m_{\ell}(L_{\ell}) = m_0(L_0)$ for all $i \neq \ell$.

**Proof.** Valid matches means that the transformations $G \xrightarrow{p_{i\ell}, m_{i\ell}} G_i$ are well-defined. In graphs and graph-like structures, $(P_{i\ell})$ is a pullback if and only if $m_i(L_i) \cap m_{\ell}(L_{\ell}) = m_0(L_0)$. Then Fact 6.1 implies that the matches form a maximal weakly disjoint matching without application conditions.

**Example 6.1.** Consider the interaction scheme $\mathcal{I} = (s_1, s_2)$ defined by the kernel morphisms $s_1$ and $s_2$ in Fig. 6, the graph $X$ depicted in the middle of Fig. 13, and the kernel rule match $m_0$ mapping the node 1 in $L_0$ to the node 1 in $X$.

If we choose maximal weakly independent matchings, the construction works as follows defining the following matches, where $f$ is the edge from 1 to 2 in $L_1$ and $g$ the reverse edge in $L_2$:

- $i = 1 : \hat{p}_1 = p_1$, $m_1 : 2 \mapsto 3, f \mapsto c$,
- $i = 2 : \hat{p}_2 = p_1$, $m_2 : 2 \mapsto 4, f \mapsto d$,
- $i = 3 : \hat{p}_3 = p_2$, $m_3 : 3 \mapsto 2, g \mapsto a$,
- $i = 4 : \hat{p}_4 = p_1$, $m_4 : 2 \mapsto 4, f \mapsto e$,
- $i = 5 : \hat{p}_5 = p_2$, $m_5 : 3 \mapsto 2, g \mapsto b$.

Thus, we find five different matches, three for the multi rule $r_1$ and two for the multi rule $r_2$. Note that in addition to the overlapping $m_0$, the matches $m_3$ and $m_5$ overlap in the node 2, while $m_2$ and $m_4$ overlap in the node 4. But since these matches are still weakly independent, because the nodes 2 and 4 are not deleted by the rule applications, this is a valid maximal weakly independent matching. It leads to the bundle $s = (s_1, s_0, s_1, s_2, s_2)$ and the amalgamated rule $\hat{p}_s$, which can be applied to $X$ leading to the amalgamated transformation $X \xrightarrow{p_{s}, m_{s}} X'$ as shown in the left of Fig. 13.
Multi-Amalgamation in $\mathcal{M}$-Adhesive Categories

If we choose maximal weakly disjoint matchings instead, the matches $m_4$ and $m_5$ are no longer valid because they overlap with $m_2$ and $m_3$, respectively, in more than the match $m_0$. Thus we obtain the maximal weakly disjoint matching $(m_0, m_1, m_2, m_3)$, the corresponding bundle $s' = (s_1, s_1, s_2)$ leading to the amalgamated rule $\tilde{p}s'$ and the amalgamated transformation $X \xrightarrow{\tilde{p}s', \tilde{m}} X''$ depicted in the right of Fig. 13. Note that this matching is not unique, also $(m_0, m_1, m_2, m_4)$ could have been chosen as a maximal weakly disjoint matching.

7. Conclusion

In this paper, we have generalized the theory of amalgamation in (Böhm et al., 1987) to multi-amalgamation in $\mathcal{M}$-adhesive categories and introduced interaction schemes and maximal matchings. More precisely, the Complement Rule and Amalgamation Theorems in (Böhm et al., 1987) 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 $\mathcal{M}$-adhesive categories for $n$ rules with application conditions (Habel and Pennemann, 2009). These generalizations are non-trivial and important for applications of parallel graph transformations to communication-based systems (Taentzer, 1996), to model transformations from BPMN to BPEL (Biermann et al., 2010a), and for the modeling of the operational semantics of visual languages (Ermel, 2006), 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 (Ehrig et al., 2006), 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 (Golas et al., 2011).

References


Appendix A. Proofs of Facts and Theorems

In this appendix, we prove the facts and theorems used within the main part. They rely on the technical lemmas proven in Appendix B.

A.1. Proof of Thm. 4.1

First, we consider the construction without application conditions.

Since $s_1$ is a kernel morphism the diagrams (11) and (21) in the right are pullbacks, and (11) has a pushout complement $(1'_1)$ for $s_{1L} \circ l_0$ (see Def. 4.1). Construct the pushout (31). Now construct the initial pushout $(41)$ over $s_{1R}$ with $b_1, c_1 \in \mathcal{M}$, then construct the pushout (51) in the below diagram on the left. We obtain an induced morphism $s_{13} : S_1 \rightarrow R_0$ with $s_{13} \circ s_{12} = b_1, s_{13} \circ s_{11} = r_0$, and $s_{13} \in \mathcal{M}$ by effective pushouts. Since (11) is a pullback Lem. B.1 implies that there is a unique morphism $l_{10} : K_1 \rightarrow L_{10}$ with $l_{10} \circ s_{1K} = u_1, u_1 \circ l_{10} = l_1$, and $l_{10} \in \mathcal{M}$ depicted in the middle. Then we can construct pushouts $(61) - (91)$ on the right as a decomposition of pushout (31) above which leads to $\overrightarrow{L_1}$ and $\overrightarrow{K_1}$ of the complement rule, and $e_{11}$ and $e_{12}$ are jointly epimorphic because $(71) + (91)$ is a pushout.

The pushout $(41)$ can be decomposed into pushouts $(101)$ and $(111)$ as shown on the left below obtaining the right hand side $\overrightarrow{K_1}$ of the complement rule. The pushback (21) can be decomposed into pushout (61) and square (121) which is a pullback by Lem. B.2 as shown in the middle. Now Lem. B.1 implies that there is a unique morphism $\overrightarrow{r_1} : \overrightarrow{K_1} \rightarrow \overrightarrow{K_1}$ in the right diagram with $\overrightarrow{r_1} \circ s_{14} = u_{13}, t_1 \circ \overrightarrow{r_1} = v_{12}$, and $\overrightarrow{r_1} \in \mathcal{M}$. 
The pushout $(7_1)$ implies that there is a unique morphism $\overline{\tau}_7 : R_{10} \to R_1$ as shown on the right, and by pushout decomposition of $(11_1) = (7_1) + (13_1)$ square $(13_1)$ is a pushout. Moreover, $(8_1) + (9_1)$ as a pushout over \( M \)-morphisms is also a pullback which completes the construction as shown below, leading to the required rule $\overline{\tau}_7 = (L_1 \xmapsto{s_{13}} K_1 \xmapsto{\tau} R_1)$ and $p_1 = p_0 \ast e, \overline{\tau}_7$ for rules without application conditions.

For the application conditions, suppose $ac_1 \cong \text{Shift}(s_{1,L}, ac_0) \land \text{L}(p_1^* \circ \text{Shift}(v_1, ac_1^*))$ for $p_1^* = (L_1 \xmapsto{v_1} L_{10} \xmapsto{v_{11}} E_1)$ with $v_1 = e_{12} \circ u_{11}$ and $ac_1^*$ on $L_{10}$. Now define $\overline{\tau}_7 = \text{Shift}(u_{11}, ac_1^*)$, which is an application condition on $L_1$. We have to show that $(p_1, ac_{p_0 \ast e, \overline{\tau}_7}) \cong (p_1, ac_1)$. By construction of the $E_1$-concurrent rule we have that $ac_{p_0 \ast e, \overline{\tau}_7} \cong \text{Shift}(s_{1,L}, ac_0) \land L(p_1^* \circ \text{Shift}(e_{12}, \overline{\tau}_7)) \cong \text{Shift}(s_{1,L}, ac_0) \land L(p_1^*, \text{Shift}(e_{12}, \text{Shift}(u_{11}, ac_1^*))) \cong \text{Shift}(s_{1,L}, ac_0) \land L(p_1^*, \text{Shift}(e_{12} \circ u_{11}, ac_1^*)) \cong \text{Shift}(s_{1,L}, ac_0) \land L(p_1^*, \text{Shift}(v_1, ac_1^*)) \cong ac_1$.

**A.2. Proof of Fact 5.1**

First we show the well-definedness of the morphisms $\tilde{l}_s$ and $\tilde{r}_s$.

Consider the colimits $(\tilde{L}_s, (t_{s,L})_{i=0,\ldots,n})$ of $(s_{i,L})_{i=1,\ldots,n}$, $(\tilde{K}_s, (t_{s,K})_{i=0,\ldots,n})$ of $(s_{i,K})_{i=1,\ldots,n}$, and $(\tilde{R}_s, (t_{s,R})_{i=0,\ldots,n})$ of $(s_{i,R})_{i=1,\ldots,n}$, with $t_{0,s} = t_{i,s} \circ s_{i,s}$ for $* \in \{L, K, R\}$ in the right diagram. Since $t_{s,L} \circ t_{s} \circ s_{i,K} = t_{s,L} \circ s_{i,L} \circ t_{0,s} = t_{0,L} \circ t_{0,s}$, we get an induced morphism $\tilde{L}_s : \tilde{K}_s \to \tilde{L}_s$ with $\tilde{l}_s \circ t_{i,K} = t_{i,L} \circ t_{i,s}$ for $i = 0,\ldots,n$. Similarly, we obtain $\tilde{r}_s : \tilde{K}_s \to \tilde{R}_s$ with $\tilde{r}_s \circ t_{i,K} = t_{i,R} \circ t_{i,s}$ for $i = 0,\ldots,n$. 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 $M$-morphisms. Since pushouts are closed under $M$-morphisms, the iterated pushout construction leads to $t_{i} \in M$. 
It remains to show that \((14_i)\) resp. \((14_i) + (1_i)\) and \((15_i)\) resp. \((15_i) + (2_i)\) in Def. 5.1 are pullbacks, and \((14_i)\) resp. \((14_i) + (1_i)\) has a pushout complement for \(t_{i,L} \circ l_i\). We prove this by induction over \(j\) for \((14_i)\) resp. \((14_i) + (1_i)\), the pullback property of \((15_i)\) follows analogously.

We prove: Let \(\bar{L}_j\) and \(\bar{K}_j\) be the colimits of \((s_i,L)_{i=1,...,j}\) and \((s_i,K)_{i=1,...,j}\), respectively. Then \((16_{ij})\) in the diagram below is a pullback with pushout complement property for all \(i = 0,\ldots, j\).

\[
\begin{array}{cccc}
K_i & \rightarrow & \bar{K}_j & \rightarrow & \bar{K}_i \\
\downarrow t_i & & \downarrow l_0 & & \downarrow l_i \\
L_i & \rightarrow & L_j & \rightarrow & L_i \\
\end{array}
\]

Basis \(j = 1\): The colimits of \(s_{1,L}\) and \(s_{1,K}\) are \(L_1\) and \(K_1\), respectively, which means that \((16_{01}) = (1_1) + (10_{11})\) and \((16_{11})\) are both pushouts and pullbacks.

Induction step \(j \rightarrow j + 1\): Construct \(\bar{L}_{j+1} = \bar{L}_j +_{\bar{L}_0} L_{j+1}\) and \(\bar{K}_{j+1} = \bar{K}_j +_{\bar{K}_0} K_{j+1}\) as pushouts in the right cube, and we have that the top and bottom faces as pushouts, the back faces as pullbacks, and by the van Kampen property also the front faces are pullbacks. Moreover, by Lem. B.3 the front faces have the pushout complement property, and by Lem. B.4 this holds also for \((16_{0j})\) and \((16_{ij})\) as compositions.

Thus, for a given \(n\), \((16_n)\) is the required pullback \((14_i)\) resp. \((14_i) + (1_i)\) with pushout complement property, using \(\bar{K}_n = K_\bar{s}\) and \(\bar{L}_n = L_\bar{s}\).

Moreover, we have pushout complements \((17_i)\) resp. \((17_i) + (1_i)\) for \(t_{i,L} \circ l_i\) as shown on the right. Since \(ac_0\) and \(ac_1\) are complement-compatible for all \(i\) we have that \(ac_i \cong \text{Shift}(s_{i,L},ac_0) \land L(p_{s_i}^*,\text{Shift}(v_i,ac_i^*))\).

For any \(ac_i^*\) it holds that \(\text{Shift}(t_{i,L},L(p_{s_i}^*,\text{Shift}(v_i,ac_i^*))) = L(p_{s_i}^*,\text{Shift}(\tilde{v},ac_i^*))\).

Thus, for any \(ac_i\) it holds that \(\text{Shift}(t_{i,L},L(p_{s_i}^*,\text{Shift}(v_i,ac_i^*))) = L(p_{s_i}^*,\text{Shift}(\tilde{v},ac_i^*))\), since all squares are pushouts by pushout-pullback decomposition and the uniqueness of pushout complements. Define \(ac_i^* := \text{Shift}(\tilde{v},ac_i^*)\) as an application condition on \(\bar{L}_0\). It follows that \(\bar{ac}_s = \bigwedge_{i=1,...,n} \text{Shift}(t_{i,L},ac_i) \cong \bigwedge_{i=1,...,n} (\text{Shift}(t_{i,L} \circ s_{i,L},ac_0) \land \text{Shift}(t_{i,L},L(p_{s_i}^*,\text{Shift}(v_i,ac_i^*))))) \cong \text{Shift}(t_{0,L},ac_0) \land \bigwedge_{i=1,...,n} L(p_{s_i}^*,\text{Shift}(\tilde{v},ac_i^*))\).

For \(i = 0\) define \(ac_{s0} = \bigwedge_{j=1,...,n} ac_j^*\), and hence \(\bar{ac}_s = \text{Shift}(t_{0,L},ac_0) \land L(p_{s_0}^*,\text{Shift}(\tilde{v},ac_{s0}^*))\) implies the complement-compatibility of \(ac_0\) and \(\bar{ac}_s\). For \(i > 0\), we have that \(\text{Shift}(t_{i,L},ac_0) \land L(p_{s_i}^*,\text{Shift}(\tilde{v},ac_i^*)) \cong \text{Shift}(t_{i,L},ac_i)\). Define \(ac_{si} = \bigwedge_{j=i,...,n} ac_j^*\), and hence \(\bar{ac}_s = \text{Shift}(t_{i,L},ac_i) \land L(p_{s_i}^*,\text{Shift}(\tilde{v},ac_i^*))\) implies the complement-compatibility of \(ac_i\) and \(\bar{ac}_s\).
A.3. Proof of Fact 5.3

From Fact 4.1 it follows that each single direct transformation \( G \xrightarrow{p_i} G_i \) can be decomposed into a transformation \( G \xrightarrow{p_0,m_0} G_0 \xrightarrow{\varphi,m_0} G_i \) with \( m_0 = m_i \circ s_{i,L} \), and since the bundle is \( s \)-amalgamable, \( m_0 = m_i \circ s_{i,L} = m_0 \) and \( G_0 := G_0 \) for all \( i = 1, \ldots, n \).

We have to show the pairwise parallel independence. From the constructions of the complement rule and the Concurrency Theorem we obtain the above diagram for all \( i = 1, \ldots, n \). For \( i \neq j \), from weakly independent matches it follows that we have a morphism \( p_{ij} \colon L_i \rightarrow D_j \) with \( f_j \circ p_{ij} = m_i \circ u_i \). It follows that \( f_j \circ p_{ij} \circ w_i = m_i \circ u_i \circ w_i = m_i \circ s_{i,L} \circ l_0 = m_0 \circ l_0 = m_j \circ s_{j,L} \circ l_0 = m_j \circ u_j \circ w_j = m_j \circ u_j \circ l_j \circ s_{j,K} = m_j \circ l_j \circ s_{j,K} = f_j \circ k_j \circ s_{j,K} \), and with \( f_j \in \mathcal{M} \) we have that \( p_{ij} \circ w_i = k_j \circ s_{j,K} \).

Now consider the pushout \((19_i) = (6_i) + (8_i)\) in comparison with object \( D_j \) and morphisms \( d_j \circ p_{ij} \) and \( x_j \circ u_{i2} \circ s_{i3} \) as shown below. We have that \( d_j \circ p_{ij} \circ l_0 \circ s_{i,K} = d_j \circ p_{ij} \circ w_i = d_j \circ k_j \circ s_{j,K} = x_j \circ r_j \circ s_{j,K} = x_j \circ u_j \circ s_{j3} \circ s_{j1} = x_j \circ u_j \circ r_0 = x_j \circ u_{i2} \circ s_{i3} \circ s_{i1} \). Now pushout \((19_i)\) induces a unique morphism \( q_{ij} \) with \( q_{ij} \circ u_{i1} = d_j \circ p_{ij} \) and \( q_{ij} \circ L_{i} \circ s_{i4} = x_j \circ u_{i2} \circ s_{i3} \).

For the parallel independence of \( G_0 \xrightarrow{\varphi,m_0} G_i \), \( G_0 \xrightarrow{\varphi,m_0} G_j \), we have to show that \( q_{ij} : L_i \rightarrow D_j \) satisfies \( f_j \circ q_{ij} = k_{00} \circ e_{i2} =: \overline{m_i} \).

With \( f_j \in \mathcal{M} \) and \( f_0 \circ d_j \circ p_{ij} = f_j \circ p_{ij} = m_i \circ u_i = f_0 \circ x_0 \) it follows that \( d_0 \circ p_{ij} = x_0 \) (**) . This means that \( \overline{f_j} \circ q_{ij} \circ u_{i1} = \overline{f_j} \circ d_j \circ p_{ij} = g_0 \circ d_0 \circ p_{ij} = g_0 \circ x_0 \circ l_0 = k_{00} \circ e_{i2} \circ u_{i1} \). In addition, we have that \( \overline{f_j} \circ q_{ij} \circ L_{i} \circ s_{i4} = \overline{f_j} \circ x_j \circ u_{i2} \circ s_{i3} = k_{00} \circ \overline{u_j} \circ u_{i2} \circ s_{i3} = k_{00} \circ \overline{u_j} \circ u_{i2} \circ s_{i3} = k_{00} \circ \overline{u_j} \circ u_{i2} \circ s_{i3} = k_{00} \circ e_{i2} \circ L_{i} \circ s_{i4} \). Since \((19_i)\) is a pushout, \( u_{i1} \) and \( L_{i} \circ s_{i4} \) are jointly epimorphic, and it follows that \( f_j \circ q_{ij} = k_{00} \circ e_{i2} \).

If \( ac_0 \) and \( ac_i \) are not complement-compatible, \( \overline{ac_i} = \text{true and trivially } \overline{ac_i} = \text{false for all } j \neq i \). Otherwise, we have that \( g_j \circ p_{ij} = \overline{ac_i} \), and with \( g_j \circ p_{ij} = \overline{ac_i} \), it follows that \( g_j \circ q_{ij} \circ u_{i1} \) is equivalent to \( \overline{ac_i} = \text{Shift}(u_{i1}, \overline{ac_i}) = \overline{ac_i} \).
A.4. Proof of Thm. 5.1

1. Synthesis. We have to show that \( \tilde{p}_s \) is applicable to \( G \) leading to an amalgamated transformation \( G \xrightarrow{\tilde{p}_s \circ \tilde{m}} H \) 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 5.1.

Then we can apply Fact 4.1 which implies the decomposition of \( G \xrightarrow{\tilde{p}_s \circ \tilde{m}} H \) into \( G \xrightarrow{\tilde{p}_s \circ m_i} G_i \xrightarrow{\tilde{q}_i} H \), where \( \tilde{q}_i \) is the (weak) complement rule of the kernel morphism \( t_i \).

Given the kernel morphisms, the amalgamated rule, and the bundle of direct transformations, we have pullbacks (1), (2), (14), (15) (see Def. 5.1), and pushouts (20), (21) (see proof of Fact 6.1) on the right.

Using Fact 5.3, we know that we can apply \( p_0 \) via \( m_0 \) leading to a direct transformation \( G \xrightarrow{\tilde{p}_s \circ m_0} G_0 \) given by pushouts (20) and (21) below. Moreover, we find decompositions of pushouts (20) and (20) into pushouts (1') and (22), resp. (21) and (23) by \( M \)-pushout back decomposition and uniqueness of pushout complements as shown in the bottom diagram below.

Since we have consistent matches, \( m_i \circ s_{i,L} = m_0 \) for all \( i = 1, \ldots, n \). Then the colimit \( \hat{L}_s \) implies that there is a unique morphism \( \hat{m} : \hat{L}_s \to G \) with \( \hat{m} \circ t_{i,L} = m_i \) and \( \hat{m} \circ t_{0,L} = m_0 \) (see (a) below).

Moreover, \( m_i \models ac_i \Rightarrow \hat{m} \circ t_{i,L} \models ac_i \Rightarrow \hat{m} \models \text{Shift}(t_{i,L}, ac_i) \) for all \( i = 1, \ldots, n \), and thus \( \hat{m} \models \text{Shift}(t_{i,L}, ac_i) \).

Weakly independent matches mean that there exist morphisms \( p_{ij} \) with \( f_j \circ p_{ij} = m_i \circ u_i \) for \( i \neq j \). Construct \( D \) as the limit of \( (d_{ij})_{i=1,\ldots,n} \) with morphisms \( d_i \). Now \( f_0 \) being a monomorphism with \( f_0 \circ d_{0,0} \circ p_{1j} = f_1 \circ p_{ij} = m_j \circ u_j = f_0 \circ x_{j0} \) implies that \( d_{0,0} \circ p_{j1} = x_{j0} \). It follows that \( d_{0,0} \circ p_{j1} \circ l_{j0} = x_{j0} \circ l_{j0} \), and together with \( d_{0,0} \circ k_i = x_{i0} \circ l_{i0} \) limit \( D \) implies that there exists a unique morphism \( r_j \) with \( d_i \circ r_j = p_{ij} \circ l_{j0}, d_i \circ r_i = k_i \), and \( d_0 \circ r_j = x_{j0} \circ l_{j0} \) (see (b)).
Similarly, $f_j$ being a monomorphism with $f_j \circ p_{i,j} \circ l_{i,0} \circ s_{i,K} = m_i \circ u_i \circ w_i = m_i \circ s_{i,L} \circ l_0 = m_0 \circ l_0 = m_j \circ s_{j,L} \circ l_0 = m_j \circ l_j \circ s_{j,K}$ implies that $p_{i,j} \circ l_{i,0} \circ s_{i,K} = k_j \circ s_{j,K}$. Now colimit $\tilde{K}_s$ implies that there is a unique morphism $\tilde{r}_j$ with $\tilde{r}_j \circ t_{i,K} = p_{i,j} \circ l_{i,0}$, $\tilde{r}_j \circ t_{j,K} = k_j$, and $\tilde{r}_j \circ t_{0,K} = k_j \circ s_{j,K}$ (see (c) above). Since $d_{i,0} \circ \tilde{r}_i \circ t_{i,K} = d_{i,0} \circ k_i = q_i \circ l_{i,0} = d_{j,0} \circ p_{i,j} \circ l_{i,0} = d_{j,0} \circ \tilde{r}_j \circ t_{i,K}$ and $d_{i,0} \circ \tilde{r}_i \circ t_{0,K} = d_{i,0} \circ k_i \circ s_{i,K} = k_0 = d_{j,0} \circ \tilde{r}_j \circ t_{0,K}$ colimit $\tilde{K}_s$ implies that for all $i, j$ we have that $d_{i,0} \circ \tilde{r}_i = d_{j,0} \circ \tilde{r}_j =: r$. From limit $D$ it now follows that there exists a unique morphism $\tilde{k}$ with $d_i \circ \tilde{k} = \tilde{r}_i$ and $d_0 \circ \tilde{k} = \tilde{r}$ (see (d)).

We have to show that (20s) in the left of the right diagram with $f = f_0 \circ d_0$ is a pushout. With $f \circ \tilde{k} \circ t_{i,K} = f_0 \circ d_0 \circ \tilde{k} \circ t_{i,K} = f_0 \circ \tilde{r} \circ t_{i,K} = f_0 \circ d_{i,0} \circ \tilde{r}_i \circ t_{i,K} = f_0 \circ d_{i,0} \circ k_i = f_i \circ k_i = m_i \circ t_i \circ l_i \circ m_0 \circ t_{i,K}$ and $f \circ \tilde{k} \circ t_{0,K} = f_0 \circ d_{0,0} \circ k_0 = m_0 \circ l_0$. Since $f_0 \circ d_{i,0} \circ \tilde{r}_i \circ t_{0,K} = f_0 \circ d_{i,0} \circ k_i \circ s_{i,K} = f_0 \circ k_0 = m_0 \circ l_0 = m_0 \circ t_{0,K}$ and $K_i$ being colimit it follows that $f \circ \tilde{k} = \tilde{m} \circ \tilde{l}_s$, thus the square commutes.

Pushout (23) can be decomposed into pushouts (24i) and (25i) in the middle of the diagram above. Using Lem. B.5 it follows that $D_0$ is the colimit of $(x_i)_{i=1,...,n}$, because (23s) is a pushout, $D$ is the limit of $(d_{i,0})_{i=1,...,n}$, and we have morphisms $p_{i,j}$ with $d_{i,0} \circ p_{i,j} = q_i$. Then Lem. B.6 implies that also (25) on the right is a pushout, where $+$ represents the coproduct construction with index $i = 1, \ldots, n$ with injections $\iota_{K_i}$ and $\iota_{L_{0,0}}$, respectively.

Consider the $n$-ary coequalizers $K_s$ of $(\iota_{K_i} \circ s_{i,K} : K_0 \to +K_i)_{i=1,...,n}$ (which is actually $\tilde{K}_s$ by construction of colimits), $\tilde{L}_0$ of $(\text{total}_{L_{0,0}} \circ w_i : K_0 \to +L_{i,0})_{i=1,...,n}$ (as already constructed in Fact 5.1), $D$ of $(\tilde{k} \circ t_{0,K} : K_0 \to D)_{i=1,...,n}$, and $D_0$ of $(k_0 : K_0 \to D_0)_{i=1,...,n}$. In the right cube, the top square with identical morphisms is a pushout, the top cube commutes, and the middle square is pushout (25) from above. Using Lem. B.7 it follows that also the bottom square (26) constructed of the four coequalizers is a pushout.

Now consider the cube below, where the top and middle squares are pushouts and the two top cubes commute. Using again Lem. B.7 it follows that (20s) in the bottom is actually a pushout, where $(27) = (1')_t + (17)_i$ is a pushout by composition. Now we can construct pushout (21s) which completes the direct transformation $G \overset{\tilde{p}_s,\tilde{m}}{\longrightarrow} H$. 
2. Analysis. Using the kernel morphisms \( t_i \) we obtain transformations \( G \xrightarrow{p_i \cdot m_i} G_i \xrightarrow{q_i} H \) from Fact 4.1 with \( m_i = \tilde{m} \circ t_{i,L} \). We have to show that this bundle of transformation is \( s \)-amalgamable. Applying again Fact 4.1 we obtain transformations \( G \xrightarrow{p_0 \cdot m_0} G_0 \xrightarrow{\phi} G_i \) with \( m_0 = m_i \circ s_{i,L} \). It follows that \( m_0 = m_i \circ s_{i,L} = \tilde{m} \circ t_{i,L} \circ s_{i,L} = \tilde{m} \circ t_{0,L} = \tilde{m} \circ t_{j,L} \circ s_{j,L} = m_j \circ s_{j,L} \) and thus we have consistent matches with \( m_0 := m_0^0 \) well-defined and \( G_0 = G_0^0 \).

It remains to show the weakly independent matches. Given the above transformations we have pushouts \((20_0), (20), (20_s)\) as above. Then we can find decompositions of \((20_0)\) and \((20_s)\) into pushouts \((27)+(28)\) and \((26)+(28)\), respectively, as shown on the right. Using pushout \((26)\) and Lem. B.8 it follows that \((25)\) as above is a pushout, since \( K_s \) is the colimit of \((s_{i,L})_{i=1,\ldots,n}\) and \( L_0 \) is the colimit of \((x_{i,L})_{i=1,\ldots,n}\) and \( l_{K_0} \) is obviously an epimorphism.

Now Lem. B.6 implies that there is a decomposition into pushouts \((24_i)\) with colimit \( D_0 \) of \((x_{i,L})_{i=1,\ldots,n}\) and pushout \((25_i)\) by \( \mathcal{M} \)-pushout pullback decomposition depicted below. Since \( D_0 \) is the colimit of \((x_{i,L})_{i=1,\ldots,n}\) and \((25_i)\) is a pushout it follows that \( D_j \) is the colimit of \((x_{i,L})_{i=1,\ldots,j-1,j+1,\ldots,n}\) with morphisms \( q_{ij} : P_i \rightarrow D_j \) and \( d_{ij} = q_{ij} \). Thus we obtain for all \( i \neq j \) a morphism \( p_{ij} = q_{ij} \circ x_{i0} \) and \( f_j \circ p_{ij} = f_0 \circ d_{ij} \circ q_{ij} \circ x_{i0} = f_0 \circ y_{i0} \circ x_{i0} = m_i \circ u_i \).

3. Bijective Corresponence. Because of the uniqueness of the used constructions, the above constructions are inverse to each other up to isomorphism.
Appendix B. Additional Lemmas

The following lemmas are valid in all adhesive and $\mathcal{M}$-adhesive categories and used in the proofs of the main theorems, where Lemmas B.1 and B.2 are used in the proof of Thm. 4.1, Lemmas B.3 and B.4 in the proof of Fact 5.1, and Lemmas B.5, B.6, B.7, and B.8 in the proof of Thm. 5.1.

Lemma B.1 ($\mathcal{M}$ complement property). If (1) is a pushout, (2) is a pullback, and $n' \in \mathcal{M}$ then there exists a unique morphism $c : C' \to C$ such that $c \circ f' = f$, $n \circ c = n'$, and $c \in \mathcal{M}$.

Proof. Since (2) is a pullback, $n' \in \mathcal{M}$ implies that $m \in \mathcal{M}$, and then also $n \in \mathcal{M}$ because (1) is a pushout. Construct the pullback (3) with $v, v' \in \mathcal{M}$, and since $n' \circ f' = g \circ m = n \circ f$ there is a unique morphism $f^* : A \to C''$ with $v \circ f^* = f'$ and $v' \circ f^* = f$. Now consider the following cube (4), where the bottom face is pushout (1), the back left face is a pullback because $m \in \mathcal{M}$, the front left face is pullback (2), and the front right face is pullback (3). Now by pullback composition and decomposition also the back right face is a pullback, and then the VK property implies that the top face is a pushout. Since (5) is a pushout and pushout objects are unique up to isomorphism this implies that $v$ is an isomorphism and $C'' \cong C'$. Now define $c := v' \circ v^{-1}$ and we have that $c \circ f' = v' \circ v^{-1} \circ f' = v' \circ f^* = f$, $n \circ c = n \circ v' \circ v^{-1} = n'$, and $c \in \mathcal{M}$ by decomposition of $\mathcal{M}$-morphisms.

Lemma B.2 ($\mathcal{M}$ pullback-pushout decomposition). If (1) + (2) is a pullback, (1) is a pushout, (2) commutes, and $o \in \mathcal{M}$ then also (2) is a pullback.

Proof. With $o \in \mathcal{M}$, (1) + (2) being a pullback, and (1) being a pushout we have that $m, n \in \mathcal{M}$. Construct the pullback (3) of $o$ and $g'$, it follows that $\pi \in \mathcal{M}$ and we get an induced morphism $b : B \to \overline{B}$ with $\overline{g} \circ b = g$, $\pi \circ b = n$, and by decomposition of $\mathcal{M}$-morphisms $b \in \mathcal{M}$.

By pullback decomposition, also (4) is a pullback and we can apply Lem. B.1 with pushout (1) and $\overline{\pi} \in \mathcal{M}$ to obtain a unique morphism $\overline{b} \in \mathcal{M}$ with $n \circ \overline{b} = \pi$ and $\overline{b} \circ b \circ f = f$. Now $n \in \mathcal{M}$ and $n \circ \overline{b} \circ b = \pi \circ b = n$ implies that $\overline{b} \circ b = id_B$, and similarly
\(\pi \in \mathcal{M}\) and \(\pi \circ b \circ b = n \circ b = \pi\) implies that \(b \circ b = id_{\pi}\), which means that \(B\) and \(\overline{B}\) are isomorphic such that also (2) is a pullback.

**Lemma B.3.** Given the following commutative cube with the bottom face as a pushout, then the front right face has a pushout complement over \(g \circ b\) if the back left face has a pushout complement over \(f \circ a\).

**Proof.** Construct the initial pushout (1) over \(f\).

Since the back left face has a pushout complement there is a morphism \(b^* : B_f \to A'\) such that \(a \circ b^* = b_f\). Since the bottom face is a pushout, (2) as the composition is the initial pushout over \(g\). Now \(b \circ m' \circ b^* = m \circ a \circ b^* = m \circ b_f\), and thus the pushout complement of \(g \circ b\) exists.

**Lemma B.4.** Given pullbacks (1) and (2) with pushout complements over \(f' \circ m\) and \(g' \circ n\), respectively, then also (1) + (2) has a pushout complement over \((g' \circ f') \circ m\).

**Proof.** Let \(C'\) and \(E'\) be the pushout complements of (1) and (2), respectively. By Lem. B.1 there are morphisms \(c\) and \(e\) such that \(c \circ f = f^*, n^* \circ e = n, e \circ g = g^*,\) and \(o^* \circ e = o\).

Now (2') can be decomposed into pushouts (3) and (4), and (1') + (4) is also a pushout and the pushout complement of \((g' \circ f') \circ m\).
Lemma B.5. Given the pushouts (1) and (3) with \( b_i \in \mathcal{M} \) for \( i = 1, \ldots, n \), morphisms \( f_{ij} : B_i \rightarrow C_j \) with \( c_j \circ f_{ij} = d_i \) for all \( i \neq j \), and the limit (2) of \( (c_j)_{j=1,\ldots,n} \) such that \( g_i \) is the induced morphism into \( E \) with \( c_i \circ g_i = a_i \) and \( e_j \circ g_i = f_{ij} \circ b_i \) using \( c_j \circ f_{ij} \circ b_i = d_i \circ b_i = c_i \circ a_i \), then (4) is the colimit of \( (h_i)_{i=1,\ldots,n} \), where \( l_i \) is the induced morphism from pushout (3) compared with \( \tau \circ g_i = c_i \circ e_i \circ g_i = c_i \circ a_i = d_i \circ b_i \).

Proof. We prove this by induction over \( n \).

I.B. \( n = 1 \): For \( n = 1 \), we have that \( C_1 \) is the limit of \( c_1 \), i.e. \( E = C_1 \), it follows that \( F_1 = C_1 \) for the pushout (3) = (1), and obviously (4) is a colimit.

I.S. \( n \rightarrow n+1 \): Consider the pushouts (1) with \( b_i \in \mathcal{M} \) for \( i = 1, \ldots, n+1 \), morphisms \( f_{ij} : B_i \rightarrow C_j \) with \( c_j \circ f_{ij} = d_i \) for all \( i \neq j \), the limits (2) of \( (c_i)_{i=1,\ldots,n} \) and \( (c_i)_{i=1,\ldots,n+1} \), respectively, leading to pullback (5) by construction of limits. Moreover, \( g_n \) and \( g_{n+1} \) are the induced morphisms into \( E_n \) and \( E_{n+1} \), respectively, leading to pushouts (3) and (3). By induction hypothesis, (4) is the colimit of \( (h_i)_{i=1,\ldots,n} \), and we have to show that (4) is the colimit of \( (h_i)_{i=1,\ldots,n+1} \).

Since (2) is a limit and \( c_i \circ f_{n+1} = d_{n+1} \) for all \( i = 1, \ldots, n \), we obtain a unique morphism \( m_{n+1} \) with \( c_n \circ m_{n+1} = f_{n+1} \) and \( e_n \circ m_{n+1} = d_{n+1} \). Since (1) is a pushout and (5) is a pullback, by \( \mathcal{M} \)-pushout-pullback decomposition also (5) and (6) are pushouts, and it follows that \( F_{n+1} = E_n \). From pushout (3) and \( h_n \circ p_{n+1} \circ g_{n+1} = h_n \circ g_n = k_n \circ b_n \) we get an induced morphism \( q_{n+1} \) with \( q_{n+1} \circ h_{n+1} = h_n \circ p_{n+1} \circ g_n \circ k_{n+1} = k_n \), and from pushout decomposition also (7) is a pushout.
To show that \((4_{n+1})\) is a colimit, consider an object \(X\) and morphisms \((x_i)\) and \(y\) with \(x_i \circ h_{n+1} = y\) for \(i = 1, \ldots, n\) and \(x_{n+1} \circ p_{n+1} = y\). From pushout \((7_{n+1})\) we obtain a unique morphism \(z\) with \(z \circ q_{n+1} = x_i\) and \(z \circ h_{n+1} = x_{n+1}\). Now colimit \((4_n)\) induces a unique morphism \(z\) with \(z \circ e_n = x_{n+1}\) and \(z \circ l_{n+1} = z_i\). It follows directly that \(z \circ l_{n+1} = z \circ l_{n+1} \circ q_{n+1} = z_i \circ q_{n+1} = x_i\) and \(z \circ e_n = z \circ e_n \circ p_{n+1} = x_{n+1} \circ p_{n+1} = y\). The uniqueness of \(z\) follows directly from the construction, thus \((4_{n+1})\) is the required colimit.

**Lemma B.6.** Given the following diagrams \((1_i)\) for \(i = 1, \ldots, n\), \((2)\), and \((3)\), with \(b = +b_i\), and \(a\) and \(e\) induced by the coproducts +\(A_i\) and +\(B_i\), respectively, then we have:

1. If \((1_i)\) is a pushout and \((2)\) a colimit then also \((3)\) is a pushout.
2. If \((3)\) is a pushout then we find a decomposition into pushout \((1_i)\) and colimit \((2)\) with \(e_i \circ d_i = e \circ i_{B_i}\).

**Proof.** 1. Given an object \(X\) and morphisms \(y, z\) with \(y \circ a = z \circ b\). From pushout \((1_i)\) we obtain with \(z \circ i_{B_i} \circ b_i = z \circ b \circ i_{A_i} = y \circ a \circ i_{A_i} = y \circ a_i\) a unique morphism \(x_i\),

2. Given an object \(X\) and morphisms \(y, z\) with \(y \circ a = z \circ b\). From pushout \((1_i)\) we obtain with \(z \circ i_{B_i} \circ b_i = z \circ b \circ i_{A_i} = y \circ a \circ i_{A_i} = y \circ a_i\) a unique morphism \(x_i\),

3. Given an object \(X\) and morphisms \(y, z\) with \(y \circ a = z \circ b\). From pushout \((1_i)\) we obtain with \(z \circ i_{B_i} \circ b_i = z \circ b \circ i_{A_i} = y \circ a \circ i_{A_i} = y \circ a_i\) a unique morphism \(x_i\).
with \( x_i \circ c_i = y \) and \( x_i \circ d_i = z \circ i_{B_i} \). Now colimit (2) implies a unique morphism \( x \) with \( x \circ c = y \) and \( x \circ c_i = x_i \). It follows that \( x \circ c \circ i_{B_i} = x \circ c_i \circ d_i = x_i \circ d_i = z \circ i_{B_i} \), and since \( z \) is unique w.r.t. \( z \circ i_{B_i} \) it follows from the coproduct that \( z = x \circ e \). Uniqueness of \( x \) follows from the uniqueness of \( x \) and \( x_i \), and hence (3) is a pushout.

2. Define \( (\iota_i := a \circ i_{A_i}) \). Now construct pushout (1\(_i\)). With \( e \circ i_{B_i} \circ b_i = e \circ o \circ i_{A_i} = e \circ a_i \) pushout (1\(_i\)) induces a unique morphism \( e_i \) with \( e_i \circ d_i = e \circ i_{B_i} \) and \( e_i \circ c_i = c_i \).

Given an object \( X \) and morphisms \( y, y_i \) with \( y_i \circ c_i = y \) we obtain a morphism \( z \) with \( z = y_i \circ d_i \) from coproduct \( + \) \( B_i \). Then we have that \( y \circ a \circ i_{A_i} = y_i \circ c_i \circ a_i = y_i \circ d_i \circ b_i = z \circ i_{B_i} \circ b_i \circ a_i \), and from coproduct \( + \) \( A_i \) it follows that \( y \circ a = z \circ b \).

Now pushout (3) implies a unique morphism \( x \) with \( x \circ c = y \) and \( x \circ c_i = x_i \). From pushout (1\(_i\)) using \( x \circ c_i \circ d_i = x \circ o \circ i_{B_i} = z \circ i_{B_i} = y_i \circ d_i \) and \( x \circ c_i \circ c_i = x \circ c = y = y_i \circ c_i \) it follows that \( x \circ c_i = y_i \), thus (2) is a colimit.

**Lemma B.7.** Consider colimits (1) – (4) such that (5\(_i\)) is a pushout for all \( i = 1, \ldots, n \) and (6\(_i\)) – (\( \iota_i \)) commute for all \( k = 1, \ldots, m \). Then also (10) is a pushout.

**Proof.** The morphisms \( T_f, \overline{g}, \overline{h}_i \) and \( \overline{K} \) are uniquely induced by the colimits. We show this exemplarily for the morphism \( T_f \). From colimit (1), with \( \overline{h}_j \circ f_j \circ a_k = \overline{b}_j \circ b_k \circ f_i = \overline{b}_j \circ f_j \) we obtain a unique morphism \( T \) with \( T \circ \overline{T} = \overline{T} \circ T_i \). It follows directly that \( \overline{K} \circ \overline{T} = \overline{T} \circ \overline{K} \).

Now consider an object \( X \) and morphisms \( y, z \) with \( y \circ \overline{g} = z \). From pushout (5\(_i\)) with \( y \circ \overline{g} \circ g_i = y \circ \overline{g} \circ \overline{a}_i = z \circ \overline{f} \circ \overline{a}_i = z \circ \overline{b}_i \circ f_i \) we obtain a unique morphism \( x_i \) with \( x_i \circ k_i = y \circ \overline{a}_i \) and \( x_i \circ h_i = z \circ \overline{b}_i \).
For all $k = 1, \ldots, m$, $x_j \circ d_k \circ k_i = x_j \circ k_j \circ c_k = y \circ \overline{c}_j \circ c_k = y \circ \overline{c}_i$ and $x_j \circ d_k \circ h_i = x_j \circ h_j \circ b_k = z \circ \overline{b}_j \circ b_k = z \circ \overline{b}_i$, and pushout \((5)\) implies that $x_i = x_j \circ d_k$. This means that colimit \((4)\) implies a unique $x$ with $x \circ \overline{d}_i = x_i$. Now consider colimit \((2)\), and $x \circ \overline{b}_i \circ \overline{a}_i = x \circ \overline{d}_i \circ h_i = x_i \circ h_i = z \circ \overline{b}_i$ implies that $x \circ \overline{b} = z$. Similarly, $x \circ \overline{b} = y$, and the uniqueness follows from the uniqueness of $x$ with respect to \((4)\). Thus, \((10)\) is indeed a pushout.

**Lemma B.8.** Consider colimits \((1)\) and \((2)\) such that \((3)\) commutes for all $i = 1, \ldots, n$, $f$ is an epimorphism, and \((4)\) is a pushout induced by colimit \((1)\). Then also \((5)\) is a pushout, where $c$ and $d$ are induced from the coproducts.

**Proof.** Since \((1)\) is a colimit and $\overline{b}_i \circ f_i \circ a_i = \overline{b}_i \circ b_i \circ f = \overline{b} \circ f$, we actually get an induced $\overline{f}$ with $\overline{f} \circ \overline{a}_i = \overline{b}_i \circ f_i$ and $\overline{f} \circ \overline{a} = \overline{b} \circ f$. From the coproducts, we obtain induced morphisms $c$ with $c \circ A_i = \overline{c} \circ \overline{a}_i$ and $d$ with $d \circ B_i = \overline{d} \circ b_i$. Moreover, for all $i = 1, \ldots, n$ we have that $d \circ (f_i) \circ i_{A_i} = d \circ i_{B_i} \circ f_i = \overline{d} \circ b_i \circ f_i = \overline{d} \circ \overline{f} \circ \overline{a}_i = \overline{c} \circ \overline{a}_i$. Uniqueness of the induced coproduct morphisms leads to $d \circ (f_i) = c \circ A_i$, i.e. \((5)\) commutes.

We have to show that \((5)\) is a pushout. Given morphisms $x, y$ with $x \circ c = y \circ (f_i)$, we have that $y \circ i_{B_i} \circ b_i \circ f = y \circ i_{B_i} \circ f_i \circ a_i = y \circ (f_i) \circ i_{A_i} \circ a_i = x \circ \overline{c} \circ \overline{a}_i \circ a_i = x \circ \overline{c} \circ \overline{a}_i$ for all $i = 1, \ldots, n$, $f$ being an epimorphism implies that $y \circ i_{B_i} \circ b_i = y \circ i_{B_i} \circ b_j$ for all $i, j$. Now define $y' := y \circ i_{B_i} \circ b_i$, and from colimit \((2)\) we obtain a unique morphism $\overline{y}$ with $\overline{y} \circ \overline{b}_i = y \circ i_{B_i}$ and $\overline{y} \circ \overline{b} = y'$. Now $x \circ \overline{c} \circ \overline{a}_i = x \circ \overline{c} \circ i_{A_i} = y \circ (f_i) \circ i_{A_i} = y \circ i_{B_i} \circ f_i = \overline{y} \circ \overline{b}_i \circ f_i = \overline{y} \circ \overline{f} \circ \overline{a}_i$ and $x \circ \overline{c} \circ \overline{a}_i = x \circ \overline{c} \circ \overline{a}_i \circ a_i = y \circ \overline{f} \circ \overline{a}_i = \overline{y} \circ \overline{f} \circ \overline{a}_i$, and the uniqueness of the induced colimit morphism implies that $\overline{y} \circ \overline{f} = x \circ \overline{c}$. This means that $X$ can be compared to pushout \((4)\), and we obtain a unique morphism $z$ with $z \circ \overline{d} = \overline{y}$ and $z \circ e = x$. Now $z \circ d \circ i_{B_i} = z \circ \overline{d} \circ \overline{b}_i = \overline{y} \circ \overline{b}_i = y \circ i_{B_i}$, and it follows that $z \circ d = y$. Similarly, the uniqueness of $z$ w.r.t. the pushout property of \((5)\) follows, thus \((5)\) is a pushout. \qed