Multi-amalgamation of rules with application conditions in \( M \)-adhesive categories

ULRIKE GOLAS\(^\dagger\), ANNEGRET HABEL\(^\ddagger\) and HARTMUT EHRIG\(^\S\)

\(^\dagger\)Konrad-Zuse-Zentrum für Informationstechnik Berlin, Berlin, Germany
Email: golas@zib.de
\(^\ddagger\)Universität Oldenburg, Oldenburg, Germany
Email: annegret.habel@informatik.uni-oldenburg.de
\(^\S\)Technische Universität Berlin, Berlin, Germany
Email: ehrig@cs.tu-berlin.de

Received 5 September 2011; revised 30 December 2011

Amalgamation is a well-known concept for graph transformations that is used to model synchronised parallelism of rules with shared subrules and corresponding transformations. This concept is especially important for an adequate formalisation of the operational semantics of statecharts and other visual modelling 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 so far only been developed on a set-theoretical basis for pairs of standard graph rules without any application conditions.

For this reason, in the current paper we present the theory of amalgamation for \( M \)-adhesive categories, which form 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, which shows how to construct a minimal complement rule for each subrule, and the Multi-Amalgamation Theorem, which generalises the well-known Parallelism and Amalgamation Theorems to the case of multiple synchronised parallelism. In order to apply the largest amalgamated rule, we use maximal matchings, which are computed according to 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 an introductory example and to provide motivation.

1. Introduction and related work

1.1. Historical background for amalgamation

The concepts of adhesive (Lack and Sobociński 2005) and weak adhesive high-level replacement (HLR) (Ehrig et al. 2006) categories were a breakthrough for the double-pushout approach of algebraic graph transformations (Rozenberg 1997). Almost all the main results for graph transformation systems could be formulated and proved in these categorical frameworks and instantiated to a large variety of HLR systems, including various kinds of graph and Petri net transformation systems (Ehrig et al. 2006). These
results included the Local Church–Rosser, Parallelism and Concurrency Theorems, the Embedding and Extension Theorem, the completeness of critical pairs, and the Local Confluence Theorem (Ehrig et al. 2014). Ehrig et al. (2010) showed that $\mathcal{M}$-adhesive categories, which are a slightly weaker version, are also sufficient for formulating 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), the application of the main theorems for the analysis of such systems is also of interest. One example is the Parallelism Theorem (Ehrig and Kreowski 1976), which states that two parallel independent transformations can be combined and are equivalent to a single transformation using the corresponding parallel rule. However, a weaker form of parallel independence is often required for distributed systems: two transformations do not have to be completely parallel independent, but may overlap dependently on certain well-defined elements. This generalisation of the Parallelism Theorem is called the Amalgamation Theorem, where the assumption of parallel independence is dropped and some synchronisation takes place. It was developed in Böhm et al. (1987) on a set-theoretical basis for a pair of standard graph rules without application conditions.

The synchronisation of two rules $p_1$ and $p_2$ is expressed by a common subrule $p_0$, which we call the kernel rule in the current paper. The subrule concept is formalised 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 $p_0$ to give an amalgamated rule $\tilde{p}$ representing the synchronised effects of $p_1$ and $p_2$. Now, two transformations via $p_1$ and $p_2$ are amalgamable if they are parallel independent except for the elements matched by the kernel rule. In this case, and in a similar way to the Parallelism Theorem, the two transformations can be combined and are equivalent to a single 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 us 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 $p_0 \ast_E \overline{p}$ equal to $p$. The Concurrency Theorem then allows us to decompose each transformation $G \Rightarrow H$ via $p$ into sequences $G \Rightarrow G_i \Rightarrow H$ via $p_0$ and $\overline{p}$. Moreover, an amalgamated transformation can also be sequentialised in this way.

1.2. Other parallel models of computation in graph transformation

Parallel rewriting was first studied at the level of strings. Motivated by examples from biology, ‘Lindenmayer Systems’, or L-systems for short, were developed as a mathematical theory of parallel languages in the 1970s. The main idea of L-systems is to simultaneously replace all letters of a string according to a given set of rules. This idea was generalised to graphs, which led to various kinds of parallel graph grammars and graph-L-systems (Rozenberg and Lindenmayer 1976).
There are several other graph transformation based approaches and tools that realise 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 pragmatically. Object nodes are identified to be, optionally, matched 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.

Other approaches that realise amalgamated graph transformation are AToM³, GReAT and GROOVE. Of these, AToM³ supports the explicit definition of interaction schemes in different rule editors (de Lara et al. 2004), while GROOVE implements rule amalgamation based on nested graph predicates (Rensink and Kuperus 2009). Although nesting extends the expressiveness of these transformations, writing and understanding these predicates is a fairly complicated task, and it seems to be difficult to relate them to or integrate them in the theoretical results for graph transformation. By contrast, 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).

Grønmo et al. (2009) adopted a related conceptual approach, which aimed at the transformation of collections of similar subgraphs. In that work, all the collection operators (multi-objects) in a rule are replaced by the mapped number of collection match copies. Similarly, Hoffmann et al. (2006) defined a cloning operator, where cloned nodes roughly correspond to multi-objects.

However, none of these approaches investigated the formal analysis of amalgamated graph transformation.

1.3. Applications of amalgamation

The concepts of amalgamation were applied to communication based systems in Taentzer and Beyer (1994), Taentzer (1996) and Ermel (2006), and transferred to the single-pushout approach of graph transformation in Löwe (1993). Amalgamation was used in Biermann et al. (2010a) to define a model transformation that translates simple business process models written in the Business Process Modelling Notation (BPMN) to executable processes formulated in the Business Process Execution Language for Web Services (BPEL). Amalgamation also plays a key role in the modelling of the operational semantics for visual languages (Ermel 2006). Golas et al. (2011) and Golas (2011) presented a complex case study for the operational semantics of statecharts based on typed attributed graphs and multi-amalgamation. An advantage of amalgamation is that 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 that is not only visual and intuitive, but also allows us to show termination and forms a solid basis for applying further graph transformation based analysis techniques.

The theory of amalgamation presented in the current 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 (Biermann et al. 2010c).
1.4. The aim of the current paper

In most applications, we need amalgamation for \( n \) rules (called multi-amalgamation), which are based not only on standard graph rules, but on various 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 elaborated. The main idea of the current paper is to fill this gap between theory and applications. To this end, we have developed the theory of multi-amalgamation for \( \mathcal{M} \)-adhesive systems based on rules with nested application conditions. Ehrig et al. (2014) gives a brief description of the amalgamation of exactly two rules in this framework. Our work in the current paper allows us to instantiate the theory to a large variety of graphs and corresponding graph transformation systems, and, using weak adhesive HLR categories, to typed attributed graph transformation systems (Ehrig et al. 2006) as well.

The work in the current paper extends Golas et al. (2010) in several ways. First, we consider amalgamated transformations in any \( \mathcal{M} \)-adhesive category, while Golas et al. (2010) only used adhesive categories. Second, we present the firing semantics of Petri nets as a new case study. This semantics is much smaller and easier to survey than the semantics of statecharts in Golas et al. (2011), but 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 allows us to compute the maximal amalgamated rule applicable at a specific kernel match.

1.5. Organisation of the paper

In Section 2, we discuss how to define the semantics of Petri nets using graph transformation and show that amalgamation makes it easier to define rules without the need for any additional control structure. In Section 3, we review basic notions related to \( \mathcal{M} \)-adhesive categories, transformations and application conditions. In Section 4, we introduce kernel rules, multi-rules and kernel morphisms, which lead to the Complement Rule Theorem as our first main result. In Section 5, we construct multi-amalgamated rules and transformations, and then show the Multi-Amalgamation Theorem as our second main result. Maximal matchings, which are used to compute the maximal amalgamated rule, are constructed in Section 6. Finally, we present a summary of our results and discuss future work in Section 7. All the more complex proofs are collected together in Appendix A, while some technical lemmas underlying these proofs are relegated to Appendix B.

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-place or post-place, respectively. Note that for simplicity we do not allow a place to be both a pre-arc and a post-arc of the same transition. A number of tokens is put on each place, and there is no limit on the number...
Multi-amalgamation of rules with application conditions in $\mathcal{M}$-adhesive categories

Fig. 1. (Colour online) The firing of the transition $t$

$$\text{fire}_{a_1, \ldots, a_m, b_1, \ldots, b_n} :$$

$$x_1 \ldots m\text{-times} \quad x_m$$

$$a_1 \ldots a_m$$

$$b_1 \ldots n\text{-times} \quad y_1 \ldots y_n$$

$$\Rightarrow$$

$$x_1 - a_1 \ldots x_m - a_m$$

$$a_1 \ldots a_m$$

$$b_1 \ldots b_n$$

$$y_1 + b_1 \ldots y_n + b_n$$

+ application conditions:
no other pre- and post-places
$a_i \leq x_i$ for $i = 1, \ldots, m$

Fig. 2. (Colour online) The rule scheme for firing an arbitrary transition in place/transition nets

of tokens allowed. Natural numbers at the arcs mark how many tokens are moved when the transition fires. Note that the absence of a number at an arc is an abbreviation for 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 then added to each post-place (see Figure 1). For the modelling of the nets, we use typed attributed graphs (Ehrig et al. 2006), which we will not describe in detail here. For each place, there is an attribute `token of type integer representing the number of tokens at this place. In the figures, we simply show this number inside the place.

Generally speaking, there are two main approaches in the literature for defining a rule-based semantics for models:

— In the first approach, the rules can be dependent on the actual instance of the model (Kuske et al. 2002), so there are some rule schemes or instructions that have to be applied to describe how to obtain the semantical rule for a concrete semantical step dependent on what the model instance looks like. In a place/transition net, for a transition with $m$ pre-places 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-arcs and post-arcs, this leads to a rule $\text{fire}_{a_1, \ldots, a_m, b_1, \ldots, b_n}$ (see Figure 2) describing the token handling. We then need application conditions to ensure all the pre-places and post-places are matched. In
Fig. 3. (Colour online) The general rules for firing a transition in place/transition nets

addition, we have to check that the number of tokens at a pre-place is no smaller than the corresponding arc weight. This rule scheme can be interpreted for each transition that occurs, thus defining the semantics of a concrete place/transition net. Note that to obtain all the firing rules of the 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 a model is changed, the semantical rules also have to be adapted. For arbitrary instances not known in advance, infinitely many rules appear, which are then difficult to analyse.

— For the second approach, general rules are applied according to some complex control structure (Varró 2002). For place/transition nets, we first have to mark an active transition to declare its firing (the top rule in Figure 3). Since we do not know in advance how many pre-places and post-places will need to be handled, we require one rule to delete a token from one pre-place and one rule to add a token in one post-place of a transition (the middle rules in Figure 3). Since we have to know which places have already been processed, we also have to mark these places. In the end, when all of the relevant places have been handled, first the transition and then the places can be unmarked (the bottom rules in Figure 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. In this way, all model instances are handled using the same rules. However, even for this simple example, a lot of marking is needed to ensure the correct matches. Although the single rules are relatively easy to understand, the additional helper structures, often combined with complex control structures for more difficult examples, makes it hard to understand the complete semantics.

Even for Petri nets, whose semantics can be described relatively easily on a set-theoretical basis (Reisig and Rozenberg 1998), both graph transformation approaches
discussed above have their drawbacks. When we analyse the second approach, it is obvious that the marking of the transition represents a kind of synchronisation: instead of arbitrary matches for the transition, the handling of the pre-places and post-places has to happen at the marked transition. The marking of the pre-places and post-places is required to avoid multiple processing of the same place. Neither of these markings are required for the first approach, since in that case all places are handled at the same time. Our goal is to combine both approaches to give a universal rule application for all model instances with less additional structure so that analysis becomes easier. To do this, we use amalgamation to define an interaction scheme that provides the necessary rules. The semantical step for each model instance can be computed using maximal matchings. As shown below, 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. When we use amalgamation, there is no need for infinitely many rules, which are difficult to analyse, or any control or helper structure. This makes the modelling of the semantics easier and prevents errors.

The semantics for place/transition nets using amalgamation is shown in Figure 4. The kernel rule \( p_0 \) appears twice in the top row. Note that we use rules in the double-pushout approach with a left-hand side \( L \) describing what must be found 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 and 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 synchronisation. Note that we use an application condition \( ac_0 \), shown in the middle of Figure 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 inexpensive to compute, and ensures that all pre-places and post-places of a chosen transition are mapped.

For example, the maximal weakly disjoint matching for the firing of the transition $t$ in Figure 1 with kernel match $t$ includes three matches for the multi-rule $p_1$ and two matches for the multi-rule $p_2$: one for each pre-place and post-place, respectively. Amalgamation of this maximal weakly disjoint matching leads to the amalgamated rule $\tilde{p}_s$ shown in Figure 5, which describes the complete firing of $t$. Note that this rule looks similar to an instantiation of the rule scheme in Figure 2, but is obtained by a very different construction mechanism, viz. amalgamation.

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 (Lack and Sobociński 2005; Ehrig et al. 2006), including the standard category of graphs, which we will use as a running example. $\mathcal{M}$-adhesive categories include a distinguished morphism class $\mathcal{M}$ of monomorphisms and extend adhesive categories with suitable properties: a major difference is that they only require pushouts along $\mathcal{M}$-morphisms to be vertical weak van Kampen squares.

Definition 3.1 (van Kampen square). A pushout, as at the bottom of the cube
with \( m \in \mathcal{M} \), is a vertical weak van Kampen square, or \( \mathcal{M} \)-van Kampen square for short, if it satisfies the vertical weak van Kampen property, that is, for any commutative cube where the back faces are pullbacks and the vertical morphisms \( b, c, d \in \mathcal{M} \), the top face is a pushout if and only if the front faces are pullbacks.

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

**Definition 3.2 (\( \mathcal{M} \)-adhesive category).** An \( \mathcal{M} \)-adhesive category \((\mathcal{C}, \mathcal{M})\) consists of a category \( \mathcal{C} \) and a class \( \mathcal{M} \) of monomorphisms in \( \mathcal{C} \) that is closed under isomorphisms, composition and decomposition (\( g \circ f \in \mathcal{M} \) and \( g \in \mathcal{M} \) implies \( f \in \mathcal{M} \)) such that \( \mathcal{C} \) has pushouts and pullbacks along \( \mathcal{M} \)-morphisms, \( \mathcal{M} \)-morphisms are closed under pushouts and pullbacks, and pushouts along \( \mathcal{M} \)-morphisms are \( \mathcal{M} \)-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 and \((\text{PTNets}, \mathcal{M})\) of place/transition nets, where \( \mathcal{M} \) is the class of all monomorphisms for all these examples, 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 (Ehrig et al. 2006).

In the double-pushout approach to transformations, rules give a general description of how to transform objects. The application of a rule to an object is called a transformation and is 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

\[
p = \left( L \xleftarrow{l} K \xrightarrow{r} R \right)
\]

with objects \( L, K \) and \( R \), called the 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 \( G \xrightarrow{p,m} H \):

![Diagram of a direct transformation](https://example.com/diagram.png)

**Example 3.4.** An example for a rule can be found in the top row of Figure 10. The application of the rule to the graph \( G \) leads to the transformation \( G \xrightarrow{p,m} H \) shown, 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 \( a: L \rightarrow C \), which demand a certain structure in addition to \( L \), and negative application conditions \( \neg \exists a \), forbidding such a structure. A match \( m: L \rightarrow G \) satisfies \( \exists a \) (respectively, \( \neg \exists a \)) if there is
a (respectively, no) \(M\)-morphism \(q : C \to G\) satisfying \(q \circ a = m\). More precisely, we use nested application conditions (Habel and Pennemann 2009), or just application conditions for short.

**Definition 3.5 (application condition and satisfaction).** An application condition \(ac\) over an object \(L\) is of the form

\[ac = \text{true}\]

or

\[ac = \exists (a, ac'),\]

where \(a : L \to C\) is a morphism and \(ac'\) is an application condition over \(C\). Given a condition \(ac\) over \(L\), a morphism \(m : L \to G\) satisfies \(ac\), written \(m \models ac\), if \(ac = \text{true}\) or \(ac = \exists (a, ac')\) and there exists a morphism \(q \in M\) with \(q \circ a = m\) and \(q \models ac'\):

\[
\begin{array}{ccc}
ac & \nearrow & L \\
& a & \searrow \\
& & C \\
& m & q \\
& & G
\end{array}
\]

Moreover, application conditions are closed under Boolean formulas (with finite or infinite index set) and satisfaction is extended in the usual way. To simplify the presentation, we will write \(\text{false}\) to abbreviate \(\neg \text{true}\), \(\exists a\) to abbreviate \(\exists (a, \text{true})\) and \(\forall (a, ac)\) to abbreviate \(\neg \exists (a, \neg ac)\). We will also write \(ac_C \cong ac'_C\) to denote the semantical equivalence of \(ac\) and \(ac'\) on \(C\).

**Example 3.6.** In Figure 10, the application condition \(\tilde{ac}_s\) of the rule \(\tilde{p}_s\) is stated above the rule, while the relevant morphisms are shown on the right. This condition forbids various edges coming from or going to node 1. The match morphism \(\tilde{m}\) satisfies this application condition.

In the current paper, we consider rules of the form

\[p = (L \leftarrow K \rightarrow R, ac),\]

where

\[
(L \leftarrow K \rightarrow R)
\]

is a (plain) rule and \(ac\) is an application condition on \(L\). There are two important concepts we need in order to handle rules with application conditions, namely, the shifts of application conditions over morphisms and rules (Habel and Pennemann 2009; Ehrig et al. 2014).

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

**Definition 3.7 (shift over morphism).** Given an application condition \(ac = \exists (a, ac')\) over \(P\) and a morphism \(b : P \to P'\), we define \(\text{Shift}(b, ac)\) to be an application condition over
$P'$ such that
\[
\text{Shift}(b, ac) = \bigvee_{(a', b') \in \mathcal{F}} \exists a' \left( \text{Shift}(b', ac') \right)
\]
with
\[
\mathcal{F} = \{(a', b') \mid (a', b') \in \mathcal{E}', b' \in \mathcal{M}, b' \circ a = a' \circ b\}.
\]

Moreover,
\[
\text{Shift}(b, \text{true}) = \text{true}
\]
and the construction is extended for Boolean formulas in the usual way.

**Remark 3.8.** $\mathcal{F}$ is finite if $\mathcal{E}'$ consists of jointly surjective pairs of morphisms, which is the case in our example categories.

**Example 3.9.** An example for shifting an application condition over a morphism is given on the left-hand side of Figure 7. We have that
\[
\text{Shift}(v_1, \neg \exists a') = \neg \exists a_{11},
\]
because square (*) is the only possible commuting square leading to $a_{11}$ and $b_{11}$ jointly surjective and $b_{11}$ injective.

**Fact 3.10.** Given an application condition $ac$ over $P$ and morphisms
\[
b : P \to P', \quad p : P' \to G,
\]
we have
\[
p \models \text{Shift}(b, ac)
\]
if and only if
\[
p \circ b \models a c.
\]

**Proof.** See Habel and Pennemann (2009) and Ehrig et al. (2014). \qed

By analogy with 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 3.11 (shift over rule).** Given a rule
\[ p = \left( L \leftarrow K \rightarrow R, ac \right) \]
and an application condition
\[ ac_R = \exists (a, ac'_R) \]
over \( R \), we define \( L(p, ac_R) \) to be an application condition over \( L \) with
\[ L(p, ac_R) = \exists (b, L(p^*, ac'_R)) \]
if \( a \circ r \) has a pushout complement (1) and
\[ p^* = \left( Y \leftarrow Z \rightarrow X \right) \]
is the derived rule by constructing pushout (2):
\[ LKR \]
\[ AC \]
\[ LF \]
\[ RB \]
\[ CG \]
\[ DH \]

Otherwise
\[ L(p, ac_R) = false. \]
Moreover,
\[ L(p, true) = true, \]
and the construction is extended to Boolean formulas in the usual way.

**Example 3.12.** Figure 7 gives an example of shifting an application condition over a rule shown by the two pushout squares \((PO_1)\) and \((PO_2)\) where
\[ L(p^*_1, \neg \exists a_{11}) = \neg \exists a_1. \]

**Fact 3.13.** Given a transformation \( G \xrightarrow{p,m} H \) via a rule
\[ p = \left( L \leftarrow K \rightarrow R, ac \right) \]
and an application condition \( ac_R \) over \( R \), we have
\[ m \models L(p, ac_R) \]
if and only if
\[ n \models ac_R. \]
\[ m \]
\[ n \]
\[ l \]
\[ r \]
\[ k \]
\[ y \]
\[ f \]
\[ g \]
\[ h \]

**Proof.** See Habel and Pennemann (2009).
Shifts over morphisms are compositional and shifts over morphisms and rules are compatible via double pushouts.

**Fact 3.14.** Given an application condition \( ac \) on \( R \), the double pushouts (1) and (2) and morphisms \( a, b \)

\[
\begin{array}{c}
p : L \xleftarrow{l} K \xrightarrow{r} R \xrightarrow{a} P \\
\downarrow{m} \quad \quad \downarrow{k} \quad \quad \downarrow{n} \\
p' : L' \xleftarrow{l'} K' \xrightarrow{r'} R'
\end{array}
\]

we have

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

and

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


### 3.1. General assumptions

In the rest of the paper, we assume we have an \( \mathcal{M} \)-adhesive category with binary coproducts, initial pushouts, \( \mathcal{E}'\)-\( \mathcal{M} \)-pair factorisation and effective pushouts (Ehrig *et al.* 2006; Golas 2011). We consider rules with (nested) application conditions (Habel and Pennemann 2009) as explained above, and in the presentation we will assume familiarity with parallelism and concurrency in the sense of Ehrig *et al.* (2006). Moreover, we use the corresponding constructions and results for the case with application conditions given in Ehrig *et al.* (2014). 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, which will lead us 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 gets its name from the fact that it can be applied multiple times for a given kernel rule match, as described in Section 5. We will need some more technical preconditions to ensure that the embeddings of the \( L \)-, \( K \)- and \( R \)-components and the application conditions are consistent and will allow us to construct a complement rule.
Definition 4.1 (kernel morphism). Given rules

\[
p_0 = \left( L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, ac_0 \right)
p_1 = \left( L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, ac_1 \right),
\]

a kernel morphism \( s_1 : p_0 \rightarrow p_1 \), with

\[
s_1 = (s_{1,L}, s_{1,K}, s_{1,R})
\]

consists of \( M \)-morphisms

\[
s_{1,L} : L_0 \rightarrow L_1
s_{1,K} : K_0 \rightarrow K_1
s_{1,R} : R_0 \rightarrow R_1
\]
such that (1) and (2) in the following diagram are pullbacks, (1) has a pushout complement (1') for \( s_{1,L} \circ l_0 \), and \( ac_0 \) and \( ac_1 \) are complement-compatible with respect to \( s_1 \), that is, given pushout (3), we have

\[
ac_1 \cong \text{Shift}(s_{1,L}, ac_0) \land L(p_1^*, \text{Shift}(v_1, ac'_1))
\]

for some \( ac'_1 \) on \( L_{10} \) and

\[
p_1^* = \left( L_1 \xleftarrow{u_1} L_{10} \xrightarrow{v_1} E_1 \right).
\]

In this case, \( p_0 \) is called a kernel rule and \( p_1 \) a multi-rule.

Remark 4.2. 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 we will use the latter for the application conditions of the complement rule later in the paper.

Example 4.3. To explain the concept of amalgamation, we will model a small transformation system for switching the direction of edges in labelled graphs, where we only have different labels for edges – black and dotted edges. The kernel rule \( p_0 \) is shown at the top of Figure 6. It selects a node with a black loop, deletes this loop and adds a dotted loop, all of this provided 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.

The middle and bottom rows of Figure 6 show two multi-rules \( p_1 \) and \( p_2 \), which extend the rule \( p_0 \) and also 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
Multi-amalgamation of rules with application conditions in $\mathcal{M}$-adhesive categories

$p_0 : a_0 \implies L_0 \rightarrow K_0 \rightarrow R_0 \rightarrow L_0 \rightarrow a_0$

$a_0 = \neg a_0$

$p_0 : a_0 \implies L_0 \rightarrow K_0 \rightarrow R_0 \rightarrow L_0 \rightarrow a_0$

$p_1 : a_1 \implies L_1 \rightarrow K_1 \rightarrow R_1 \rightarrow L_1 \rightarrow a_1$

$a_1 = \text{Shift}(s_1, L, a_0) \land \neg \exists a_1$

$p_2 : a_2 \implies L_2 \rightarrow K_2 \rightarrow R_2 \rightarrow L_2 \rightarrow a_2$

$a_2 = \text{Shift}(s_2, L, a_0) \land \neg \exists a_2$

Fig. 6. (Colour online) The kernel rule $p_0$ and the multi-rules $p_1$ and $p_2$

kernel morphism $s_1 : p_0 \rightarrow p_1$ as shown at the top of Figure 6 with pullbacks $(1_1)$ and $(2_1)$, and pushout complement $(1'_1)$. For the application conditions, we have

$$a_{c1} = \text{Shift}(s_1, L, a_0) \land \neg \exists a_1$$

Fig. 7. (Colour online) Constructions for the application conditions

as shown on the left-hand side of Figure 7. Thus $a_{c1}' = \neg \exists a_1'$, and $a_0$ and $a_1$ are complement compatible.
Similarly, there is a kernel morphism $s_2 : p_0 \to p_2$, as shown in the bottom row of Figure 6, with pullbacks (12) and (22), pushout complement (12'), and $ac_0$ and $ac_2$ being complement compatible.

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

**Theorem 4.4 (existence of complement rule).** Given rules

\[
p_0 = \left( L_0 \leftarrow K_0 \rightarrow R_0, ac_0 \right)
\]
\[
p_1 = \left( L_1 \leftarrow K_1 \rightarrow R_1, ac_1 \right),
\]

and a kernel morphism $s_1 : p_0 \to p_1$, there is a canonical way to construct a rule

\[
\overline{p_1} = \left( L_1 \leftarrow K_1 \rightarrow R_1, ac_1 \right)
\]

and a jointly epimorphic cospan $R_0 \xrightarrow{e_{11}} E_1 \xleftarrow{e_{12}} L_1$ such that the $E_1$-concurrent rule $p_0 *_{E_1} \overline{p_1}$ exists and $p_1 = p_0 *_{E_1} \overline{p_1}$:

See Ehrg et al. (2014) for the definition of $E$-concurrent rules for rules with application conditions.

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

**Remark 4.5.** Note that when we use 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 with respect to the property $p_1 = p_0 *_{E_1} \overline{p_1}$ since other choices for $S_1$ with $M$-morphisms $s_{11}$ and $s_{13}$ also lead to a well-defined construction. In particular, we could choose $S_1 = R_0$ leading to

\[
\overline{p_1} = E_1 \xleftarrow{\pi} R_{10} \xrightarrow{\pi} R_1.
\]

Our choice represents a smallest possible complement, which should be preferred in most application areas.
Multi-amalgamation of rules with application conditions in $\mathcal{M}$-adhesive categories

Definition 4.6 (Complement rule). Given rules

$$p_0 = \left( L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, ac_0 \right)$$
$$p_1 = \left( L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, ac_1 \right),$$

and a kernel morphism $s_1 : p_0 \to p_1$, the canonical rule

$$\overline{p_1} = \left( L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \overline{ac_1} \right)$$

identified by Theorem 4.4 is called the complement rule (of $s_1$).

Example 4.7. Consider the kernel morphism $s_1$ in Figure 6. Using the construction in Theorem 4.4, we obtain the complement rule in the top row of Figure 8 with the application condition $\overline{ac_1} = \neg \exists a_1$ constructed in the right-hand side of Figure 7. Figure 9 shows the diagrams of the construction. In a similar way, we can obtain a complement rule for the kernel morphism $s_2 : p_0 \to p_2$ in Figure 6 – see the bottom row of Figure 8.

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 4.8 (Decomposition of Transformations). Given rules

$$p_0 = \left( L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, ac_0 \right)$$
$$p_1 = \left( L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, ac_1 \right),$$

a kernel morphism $s_1 : p_0 \to p_1$, and a direct transformation $t_1 : G \xrightarrow{p_1, M_1} G_1$, we have that $t_1$ can be decomposed into the transformation

$$G \xrightarrow{p_0, M_0} G_0 \xrightarrow{\overline{p_1}, M_1} G_1$$
Fig. 9. (Colour online) The construction of the complement rule for the kernel morphism $s_1$

with

$$m_0 = m_1 \circ s_1 \circ L$$

where $\overline{p_1}$ is the complement rule of $s_1$:

Proof. We have

$$p_1 \cong p_0 \ast E_1 \overline{p_1}.$$
The analysis part of the Concurrency Theorem (Ehrig et al. 2014) then implies the decomposition into

\[ G \xrightarrow{p_0, m_0} G_0 \xrightarrow{p_1, m_1} G_1 \]

with \( m_0 = m_1 \circ s_{1,L} \).

5. Multi-amalgamation

Böhm et al. (1987) developed an Amalgamation Theorem for a pair of graph rules without application conditions, which can be seen as a generalisation of the Parallelism Theorem (Ehrig and Kreowski 1976) in which the assumption of parallel independence is dropped and pure parallelism is generalised to synchronised parallelism. In this section, we present the Multi-Amalgamation Theorem as an Amalgamation Theorem for a bundle of rules with application conditions over objects in an \( \mathcal{M} \)-adhesive category.

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

**Definition 5.1 (amalgamated rule).** Given rules

\[ p_i = \left( L_i \xleftarrow{l_i} K_i \xrightarrow{r_i} R_i, a_{c_i} \right) \]

for \( i = 0, \ldots, n \) and a bundle of kernel morphisms

\[ s = (s_i : p_0 \rightarrow p_i)_{i=1,\ldots,n}, \]

the amalgamated rule

\[ \tilde{p}_s = \left( \tilde{L}_s \xleftarrow{\tilde{l}_s} \tilde{K}_s \xrightarrow{\tilde{r}_s} \tilde{R}_s, \tilde{a}_{c_s} \right) \]

is constructed using:

- the componentwise colimits 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 \( (t_{i,R} \circ r_i)_{i=1,\ldots,n} \), respectively; and

- \( \tilde{a}_{c_s} \) given by

\[ \tilde{a}_{c_s} = \bigwedge_{i=1,\ldots,n} \text{Shift}(t_{i,L}, a_{c_i}). \]
Fact 5.2. The amalgamated rule is well defined and we have kernel morphisms

\[ t_i = (t_{i,L}, t_{i,K}, t_{i,R}) : p_i \rightarrow \tilde{p}_s \]

for \( i = 0, \ldots, n \).

Proof. See Section A.2 in the appendix.

The application of an amalgamated rule yields an amalgamated transformation.

Definition 5.3 (amalgamated transformation). The application of an amalgamated rule to a graph \( G \) is called an amalgamated transformation.

Example 5.4. Consider the bundle \( s = (s_1, s_2, s_3 = s_1) \) of kernel morphisms shown in Figure 6. The corresponding amalgamated rule \( \tilde{p}_s \) is shown in the top row of Figure 10. This amalgamated rule can be applied to the graph \( G \) leading to the amalgamated transformation shown in Figure 10, where the application condition \( \tilde{a}c_s \) 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 determine if the amalgamated rule is
applicable to $G$ in combining all of the single transformation steps. These transformations are compatible, that is, multi-amalgamable, if the matches agree on the kernel rules and are independent outside.

**Definition 5.5 (s-amalgamable).** Given a bundle of kernel morphisms

$$s = (s_i : p_0 \rightarrow p_i)_{i=1,...,n},$$

we say a bundle of direct transformations steps

$$(G \xrightarrow{p_{m_i}} G_i)_{i=1,...,n}$$

is s-amalgamable if:

— it has *consistent matches*, that is,

$$m_i \circ s_{i,L} = m_j \circ s_{j,L} =: m_0$$

for all $i, j = 1, \ldots, n$ and

— it has *weakly independent matches*, that is, for all $i \neq j$, if we consider the pushout complements $(1'_i)$ and $(1'_j)$, there exist morphisms

$$p_{ij} : L_{i0} \rightarrow D_j$$
$$p_{ji} : L_{j0} \rightarrow D_i$$

such that

$$f_j \circ p_{ij} = m_i \circ u_i$$
$$f_i \circ p_{ji} = m_j \circ u_j$$

and

$$g_j \circ p_{ij} \models ac'_i$$
$$g_i \circ p_{ji} \models ac'_j.$$

We can give a set-theoretical characterisation of weak independence without application conditions in a similar way to the characterisation of parallel independence in Ehrig *et al.* (2006).
Fact 5.6. For graphs and other set-based structures, weakly independent matching, 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 $i \neq j$, that is, 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 $i \neq j = 1, \ldots, n$ to 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') is a pushout, we have

$$y_i = u_i(z_i) \in u_i(L_0 \setminus w_1(K_0)),$$

and

$$x = m_i(u_i(z_i)) = f_j(p_{ij}(z_i)) = m_j(y_j),$$

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 $x \in L_{\partial 0}$ and $x = w_i(k)$, we define

$$p_{ij}(x) = k_j(s_{j,K}(k)).$$

Then

$$f_j(p_{ij}(x)) = f_j(k_j(s_{j,K}(k)))$$

$$= m_j(l_j(s_{j,K}(k))))$$

$$= m_j(l_j(s_{j,L}(l_0(k)))))$$

$$= m_j(u_i(w_i(k))))$$

$$= m_i(u_i(x))).$$

Otherwise, we have $x \notin w_i(K_0)$, that is, $u_i(x) \notin s_{i,L}(L_0)$, and define

$$p_{ij}(x) = y$$

with

$$f_j(y) = m_i(u_i(x))).$$
Multi-amalgamation of rules with application conditions in $\mathcal{M}$-adhesive categories

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 thus

$$m_i(u_i(x)) \in m_j(l_j(K_j)),$$

and in both cases

$$m_i(u_i(x)) \in f_j(D_j).$$

We can also define $p_{ji}$ with the required property in a similar way.

**Example 5.7.** Consider the bundle $s = (s_1, s_2, s_3 = s_1)$ of kernel morphisms we considered in Example 5.4. For the graph $G$ given in Figure 10, we find matches

$$m_0 : L_0 \to G$$
$$m_1 : L_1 \to G$$
$$m_2 : L_2 \to G$$
$$m_3 : L_1 \to G$$

mapping all nodes from the left-hand side to their corresponding nodes in $G$, except for $m_3$, which maps 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 $p_1$, $p_2$, $p_3$, respectively, to give the bundle of direct transformations shown in Figure 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 5.8.** 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
\[ (G \xrightarrow{g_i} G_i)_{i=1,...,n}, \]
each direct transformation \( G \xrightarrow{g_i} G_i \) can be decomposed into a transformation
\[ G \xrightarrow{g_0} G_0 \xrightarrow{g_i} G_i. \]
Moreover, the transformations
\[ G_0 \xrightarrow{g_0} G_i \]
are pairwise parallel independent.

*Proof.* See Section A.3 in the appendix.

If a bundle of direct transformations of a graph \( G \) is s-amalgamable, we can apply the amalgamated rule directly to \( G \) to give a parallel execution of all the changes made by the single transformation steps.

**Theorem 5.9 (multi-amalgamation).** Consider a bundle of kernel morphisms
\[ s = (s_i : p_0 \to p_i)_{i=1,...,n}. \]
Then:

1. **Synthesis:**
   Given an s-amalgamable bundle
   \[ \left( G \xrightarrow{g_i} G_i \right)_{i=1,...,n} \]
of direct transformations, there is an amalgamated transformation \( G \xrightarrow{\tilde{p}_s,\tilde{m}} H \) and transformations \( G_i \xrightarrow{q_i} H \) over the complement rules \( q_i \) of the kernel morphisms \( t_i : p_i \to \tilde{p}_s \) such that
   \[ G \xrightarrow{\tilde{p}_s,\tilde{m}} G_i \xrightarrow{q_i} H \]
is a decomposition of \( G \xrightarrow{\tilde{p}_s,\tilde{m}} H \):
Multi-amalgamation of rules with application conditions in $\mathcal{M}$-adhesive categories

(2) **Analysis:**
Given an amalgamated transformation $G \xRightarrow{\beta, \delta} H$, there are $s_i$-related transformations $G \xRightarrow{p_i, m_i} G_i \xRightarrow{q_i} H$ for $i = 1, \ldots, n$ such that $G \xRightarrow{p_i, m_i} G_i$ is $s_i$-amalgamable.

(3) **Bijective correspondence:**
The synthesis and analysis constructions are inverse to each other up to isomorphism.

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

**Remark 5.10.** 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

\begin{align*}
p_{K_0} &= \left( K_0 \xrightarrow{id_{K_0}} K_0 \xleftarrow{id_{K_0}} K_0, \text{true} \right)\end{align*}

and $\overline{p_j}$ is the complement rule of $p_j$.

For $n = 2$ and rules without application conditions, the Multi-Amalgamation Theorem specialises to the Amalgamation Theorem in Böhm et al. (1987). Moreover, if $p_0$ is the empty rule, it is just the Parallelism Theorem in Ehrig et al. (2014) since the transformations are parallel independent for an empty kernel match.

**Example 5.11.** As stated in Example 5.7, the transformations $G \xRightarrow{p_1, m_1} G_1$
$G \xRightarrow{p_2, m_2} G_2$
$G \xRightarrow{p_1, m_3} G_3$

shown in Figure 11 are $s$-amalgamable for the bundle $s = (s_1, s_2, s_3 = s_1)$

of kernel morphisms. Applying Fact 5.8, we can decompose these transformations into a transformation $G \xRightarrow{p_0, m_0} G_0$ followed by transformations $G_0 \xRightarrow{\overline{p_1}, \overline{m_1}} G_1$
$G_0 \xRightarrow{\overline{p_2}, \overline{m_2}} G_2$
$G_0 \xRightarrow{\overline{p_1}, \overline{m_3}} G_3$

via the complement rules, which are pairwise parallel independent. These transformations are shown in Figure 12. Moreover, Theorem 5.9 implies that we obtain an amalgamated transformation $G \xRightarrow{\beta, \delta} H$ for this bundle of direct transformations – this is the transformation already shown in Figure 10. Conversely, the analysis of this amalgamated
transformation leads to the $s$-amalgamable bundle of transformations

$$G \xrightarrow{p_1, m_1} G_1$$
$$G \xrightarrow{p_2, m_2} G_2$$
$$G \xrightarrow{p_3, m_3} G_3$$

from Figure 11.

6. Multi-amalgamation with maximal matchings

An important extension of the theory presented so far 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 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-places and post-places of a transition should be computed during runtime, and they are dependent on the current Petri net model.

An interaction scheme defines a bundle of kernel morphisms. In contrast to a concrete bundle, in order to apply such an interaction scheme, all possible matches for the multi-rules that agree on a given kernel match are computed and lead to an amalgamable bundle of transformations.

**Definition 6.1 (interaction scheme).** A kernel rule $p_0$ and a set of multi-rules

$$\{p_1, \ldots, p_k\}$$

with kernel morphisms $s_i : p_0 \rightarrow p_i$ form an interaction scheme

$$is = \{s_1, \ldots, s_k\}.$$
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 

\[ \text{is} = \{s_1, \ldots, s_k\} \]

with \( s_j : p_0 \to p_j \) for \( j = 1, \ldots, k \) and a family of matchings 

\[ m = (m_i : L_i' \to G), \]

where each \( p_i' \) corresponds to some \( p_j \) for \( j \leq k \), with transformations \( G \xrightarrow{p_i',m_i} G_i \), we say \( m \) forms a maximal weakly independent matching if the bundle \( G \xrightarrow{p_i',m_i} G_i \) is multi-amalgamable and, for any rule \( p_j \), no other match \( m' : L_j \to G \) can be found such that \( (m_i, m') \) fulfils this property.

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

**Algorithm 6.3 (maximal weakly independent matching).** Given a graph \( G \) and an interaction scheme 

\[ \text{is} = \{s_1, \ldots, s_k\}, \]

a maximal weakly disjoint matching 

\[ m = (m_0, m_1, \ldots, m_n) \]

can be computed as follows:

1. Set \( i = 0 \) and 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. For 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 

\[ s = (s_i : p_0 \to \hat{p}_i) \]

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 requires a lot of backtracking because a match in Step (2) is often 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 analysed 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 enough to require the matches to be disjoint outside the kernel match. A typical example is the semantics of Petri nets described in Section 2, where 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 the implementation, we can rule out model parts that have already been matched.

**Definition 6.4 (maximal weakly disjoint matching).** Given an interaction scheme

\[ is = \{s_1, \ldots, s_k\} \]

and a maximal weakly independent matching

\[ m = (m_i : L'_i \to G), \]

we say \( m \) forms a _maximal weakly disjoint matching_ if the square \((P_{\ell\ell})\) is a pullback for all \( i \neq \ell \):

\[
\begin{array}{ccc}
L_0 & \xrightarrow{s_{i,L}} & L'_i \\
\downarrow{s_{\ell,L}} & & \downarrow{m_i} \\
L'_\ell & \xrightarrow{m_\ell} & G \\
\end{array}
\]

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

**Fact 6.5.** 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 \( G \xrightarrow{p_i,m_i} G_i \) such that

\[ m_i \circ s_{i,L} = m_0 \]

and square \((P_{\ell\ell})\) is a pullback for all \( i \neq \ell \), the bundle \( G \xrightarrow{p_i,m_i} G_i \) 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, so it just remains to show that they are weakly independent.

Consider the transformations \( G \xrightarrow{p_i,m_i} G_i \) with pushouts \((20_i)\) and \((21_i)\) in the following diagram:

\[
\begin{array}{ccccccc}
& L_i & \xrightarrow{l_i} & K_i & \xrightarrow{r_i} & R_i & \\
& \downarrow{m_i} & & \downarrow{k_i} & & \downarrow{p_i} & \\
G & \xrightarrow{f_i} & D_i & \xrightarrow{g_i} & G_i & \\
\end{array}
\]
For the cube

\[
\begin{array}{c}
P \downarrow \bar{f} \quad K_0 \downarrow l_0 \\
\vert \quad \vert \\
\tilde{f} \quad L_i \downarrow \ell_i \\
\vert \quad \vert \\
L_0 \quad K_i \downarrow \ell_i \\
\vert \quad \vert \\
G \downarrow m_i \quad L_0 \end{array}
\]

the bottom face is pushout \((20)\), the back right face is pullback \((1)\) 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 since

\[
m_\ell \circ s_\ell, L \circ l_0 = m_i \circ s_i, L \circ l_0 = m_i \circ l_i \circ s_i, K = f_i \circ k_i \circ s_i, K,
\]

we obtain a morphism \(p\) with

\[
\bar{f} \circ p = s_\ell, L \circ l_0
\]

and

\[
m_\ell \circ p = f_i \circ s_i, K.
\]

From pullback composition and decomposition of the right and left faces, it follows that the back left face is a pullback too. The \(\mathcal{M}\)-van Kampen property can now be applied to give a pushout in the top face. Since pushout complements are unique up to isomorphism, we can substitute the top face by pushout \((1')\) from Definition 5.5 with \(P \cong L_{/0}\). Thus we have found the morphism \(p_{/i} := \hat{m}\) with

\[
f_i \circ p_{/i} = m_\ell \circ u_i.
\]

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 characterisation of maximal weakly disjoint matchings similar to the result in Fact 5.6.

**Fact 6.6.** 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.** The fact that we have valid matches means that the transformations

\[
G \xrightarrow{p_{/i}, m_i} G_i
\]
are well defined. In graphs and graph-like structures, \((P_\ell)\) is a pullback if and only if
\[ m_i(L_i) \cap m_\ell(L_\ell) = m_0(L_0). \]

Fact 6.5 then implies that the matches form a maximal weakly disjoint matching without application conditions.

**Example 6.7.** Consider the interaction scheme \(is = (s_1, s_2)\) defined by the kernel morphisms \(s_1\) and \(s_2\) in Figure 6, the graph \(X\) shown in the middle of Figure 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 to define the following matches, where \(f\) is the edge from 1 to 2 in \(L_1\) and \(g\) is 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 \(p_1\) and two for the multi-rule \(p_2\). 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. This leads to the bundle

\[ s = (s_1, s_1, s_2, s_2) \]

and the amalgamated rule \(\hat{p}_s\), which can be applied to \(X\) to give the amalgamated transformation \(X \xrightarrow{\hat{p}_s, m_0} X'\), as shown on the left of Figure 13.

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 \(s' = (s_1, s_1, s_2)\) giving the amalgamated rule \(\hat{p}_s'\) and the amalgamated transformation \(X \xrightarrow{\hat{p}_s', m_0'} X''\) shown on the right of Figure 13. Note that this matching is not unique, and \((m_0, m_1, m_2, m_4)\) could also have been chosen as a maximal weakly disjoint matching.
7. Conclusions

In the current paper, we have generalised the theory of amalgamation in Böhm et al. (1987) to multi-amalgamation in \( M \)-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 the current paper are valid in adhesive and \( M \)-adhesive categories for \( n \) rules with application conditions (Habel and Pennemann 2009). These generalisations are non-trivial, and are important for applications of parallel graph transformations to communication-based systems (Taentzer 1996) and to model transformations from BPMN to BPEL (Biermann et al. 2010a), and for modelling 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 for analysing interesting properties of the operational semantics, such as termination, local confluence and functional behaviour. However, generalising the corresponding results in Ehrig et al. (2006), such as the Local Church–Rosser, Parallelism and Local Confluence Theorems, to the case of multi-amalgamated rules, and, in particular, to the operational semantics of statecharts based on amalgamated graph transformation with maximal matchings in Golas et al. (2011), is left for future work.

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 proved in Appendix B.

A.1. Proof of Theorem 4.4

Proof. We begin by considering the construction without application conditions. Since \( s_1 \) is a kernel morphism, the diagrams \((1_1)\) and \((2_1)\) in

\[
\begin{array}{ccc}
L_0 & \xleftarrow{l_0} & K_0 & \xrightarrow{r_0} & R_0 \\
\downarrow{s_{1,L}} & & \downarrow{(1_1)} & & \downarrow{s_{1,R}} \\
L_1 & \xleftarrow{l_1} & K_1 & \xrightarrow{r_1} & R_1
\end{array}
\]

are pullbacks and \((1_1)\) has a pushout complement \((1'_1)\) for \( s_{1,L} \circ l_0 \) (see Definition 4.1). We construct the pushout \((3_1)\):

\[
\begin{array}{ccc}
L_0 & \xleftarrow{l_0} & K_0 & \xrightarrow{r_0} & R_0 \\
\downarrow{s_{1,L}} & & \downarrow{(1'_1)} & & \downarrow{c_{12}} \\
L_1 & \xleftarrow{u_1} & L_{10} & \xrightarrow{v_1} & E_1
\end{array}
\]
We now construct the initial pushout \((4_1)\) over \(s_{1,R}\) with \(b_1, c_1 \in \mathcal{M}\), \(P_1\) as the pullback object of \(r_0\) and \(b_1\), and the pushout \((5_1)\) in \(B_1 \rightarrow C_1 \rightarrow R_1\)

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 \((1_1)\) is a pullback, Lemma B.1 implies that there is a unique morphism \(l_{10} : K_1 \rightarrow L_{10}\) with

\[ l_{10} \circ s_{1\cdot K} = w_1 \]

\[ u_1 \circ l_{10} = l_1 \]

and \(l_{10} \in \mathcal{M}\) as in \(L_0 \rightarrow L_1 \rightarrow L_{10}\)

We can then construct pushouts \((6_1)\)–\((9_1)\)

as a decomposition of pushout \((3_1)\) above, which leads to \(L_1\) and \(K_1\) of the complement rule, so \(e_{11}\) and \(e_{12}\) are jointly epimorphic because \((7_1) + (9_1)\) is a pushout.

The pushout \((4_1)\) can be decomposed into pushouts \((10_1)\) and \((11_1)\) as in \(B_1 \rightarrow C_1 \rightarrow R_1\)
to give the right-hand side $\overline{R}_1$ of the complement rule. The pullback $(2_1)$ can be decomposed into pushout $(6_1)$ and square $(12_1)$, which is a pullback by Lemma B.2 as shown in

$$
\begin{array}{c}
K_0 \xrightarrow{s_{11}} S_1 \xrightarrow{s_{13}} R_0 \\
\downarrow s_{1, K} (6_1) \quad \downarrow s_{14} \quad \downarrow (12_1) \quad \downarrow s_{1, R} \\
K_1 \xrightarrow{u_{11}} K_1 \xrightarrow{u_{12}} R_1
\end{array}
$$

Lemma B.1 now implies that there is a unique morphism $\overline{r}_1 : \overline{K}_1 \to \overline{R}_1$ in

with

$$
\overline{r}_1 \circ s_{14} = u_{13} \\
t_1 \circ \overline{r}_1 = v_{12},
$$

and $\overline{r}_1 \in \mathcal{M}$.

The pushout $(7_1)$ implies that there is a unique morphism $\overline{v}_1 : \overline{R}_{10} \to \overline{R}_1$ as shown in

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 $\mathcal{M}$-morphisms, is also a pullback, which completes the construction shown below, leading to the required rule

$$
\overline{p}_1 = \left( L_1 \overline{r}_1 \overline{K}_1 \overline{R}_1 \right)
$$

and

$$
p_1 = p_0 *_{E_1} \overline{p}_1
$$
for rules without application conditions.

For the application conditions, suppose
\[ \text{ac}_1 \cong \text{Shift}(s_{1,L}, \text{ac}_0) \land \text{L}(p_1^*, \text{Shift}(v_1, \text{ac}_1')) \]
for
\[ p_1^* = (L_1 \xleftarrow{u_1} L_{10} \xrightarrow{v_1} E_1) \]
with
\[ v_1 = e_{12} \circ u_{11} \]
and \( \text{ac}_1' \) on \( L_{10} \). We now define
\[ \overline{\text{ac}}_1 = \text{Shift}(u_{11}, \text{ac}_1'), \]
which is an application condition on \( L_1 \). We have to show that
\[ (p_1, \text{ac}_{p_0^*E_1}) \cong (p_1, \text{ac}_1). \]
By construction of the \( E_1 \)-concurrent rule,
\[ \text{ac}_{p_0^*E_1} \cong \text{Shift}(s_{1,L}, \text{ac}_0) \land \text{L}(p_1^*, \text{Shift}(e_{12}, \overline{\text{ac}}_1)) \]
\[ \cong \text{Shift}(s_{1,L}, \text{ac}_0) \land \text{L}(p_1^*, \text{Shift}(e_{12}, \text{Shift}(u_{11}, \text{ac}_1')))) \]
\[ \cong \text{Shift}(s_{1,L}, \text{ac}_0) \land \text{L}(p_1^*, \text{Shift}(e_{12} \circ u_{11}, \text{ac}_1')) \]
\[ \cong \text{Shift}(s_{1,L}, \text{ac}_0) \land \text{L}(p_1^*, \text{Shift}(v_1, \text{ac}_1')) \]
\[ \cong \text{ac}_1. \]

A.2. Proof of Fact 5.2

Proof. We will begin by showing the well definedness of the morphisms \( \overline{I}_s \) and \( \overline{R}_s \):
Consider the colimits
\[(\tilde{L}_s, (t_i,L)_{i=0,...,n}) \quad \text{of} \quad (s_i,L)_{i=1,...,n},\]
\[(\tilde{K}_s, (t_i,K)_{i=0,...,n}) \quad \text{of} \quad (s_i,K)_{i=1,...,n},\]
\[(\tilde{R}_s, (t_i,R)_{i=0,...,n}) \quad \text{of} \quad (s_i,R)_{i=1,...,n},\]
with
\[t_{0,*} = t_{i,*} \circ s_{i,*}\]
for \(* \in \{L, K, R\}\) in the right-hand digram above. Since
\[t_{i,L} \circ l_i \circ s_{i,K} = t_{i,L} \circ s_{i,L} \circ l_0 = t_{0,L} \circ l_0,\]
we get an induced morphism \(\tilde{l}_s : \tilde{K}_s \to \tilde{L}_s\) with
\[\tilde{l}_s \circ t_{i,K} = t_{i,L} \circ l_i\]
for \(i = 0,\ldots,n\). Similarly, we obtain \(\tilde{r}_s : \tilde{K}_s \to \tilde{R}_s\) with
\[\tilde{r}_s \circ t_{i,K} = t_{i,R} \circ r_i\]
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 \(\mathcal{M}\)-morphisms. Since pushouts are closed under \(\mathcal{M}\)-morphisms, the iterated pushout construction leads to \(t_i \in \mathcal{M}\).

It remains to show that \((14_i)\) and \((14_i) + (1)_i\), and \((15_i)\) and \((15_i) + (2)_i\) in Definition 5.1 are pullbacks, and \((14_i)\) and \((14_i) + (1)_i\) have a pushout complement for \(t_{i,L} \circ l_i\). We will prove this by induction over \(j\) for the \((14_i)\) and \((14_i) + (1)_i\) case only; the pullback property for \((15_i)\) follows analogously.

Let \(\tilde{L}_j\) and \(\tilde{K}_j\) be the colimits of \((s_{i,L})_{i=1,...,j}\) and \((s_{i,K})_{i=1,...,j}\), respectively. We need to prove that \((16_{ij})\) in
\[
\begin{array}{ccc}
K_i & \xrightarrow{t_i} & \tilde{K}_j \\
\downarrow l_i & & \downarrow \ \\
L_i & \xrightarrow{(16_{ij})} & \tilde{L}_j
\end{array}
\]
is a pullback with the pushout complement property for all \(i = 0,\ldots,j\).

--- Base case \((j = 1)\):

The colimits of \(s_{1,L}\) and \(s_{1,K}\) are \(L_1\) and \(K_1\), respectively, which means that \((16_{01}) = (1)_1 + (16_{11})\) and \((16_{11})\) are both pushouts and pullbacks:
\[
\begin{array}{ccc}
K_0 & \xrightarrow{s_{1,K}} & K_1 & \xrightarrow{(16_{11})} & \tilde{K}_1 \\
\downarrow l_0 & & \downarrow l_1 & & \downarrow \ \\
L_0 & \xrightarrow{(11)} & L_1 & \xrightarrow{(16_{11})} & \tilde{L}_1
\end{array}
\]
― Induction step \((j \to j + 1)\):

We construct

\[
\begin{align*}
\tilde{L}_{j+1} &= \tilde{L}_j + L_0 L_{j+1} \\
\tilde{K}_{j+1} &= \tilde{K}_j + K_0 K_{j+1}
\end{align*}
\]

as pushouts in the cube

\[
\begin{array}{c}
\tilde{K}_j \\
\downarrow \\
\tilde{L}_j \\
\downarrow \\
L_{j+1}
\end{array}
\quad
\begin{array}{c}
K_0 \\
\downarrow \\
L_0 \\
\downarrow \\
L_{j+1}
\end{array}
\quad
\begin{array}{c}
\tilde{K}_{j+1} \\
\downarrow \\
\tilde{L}_{j+1}
\end{array}
\]

The top and bottom faces are pushouts, the back faces are pullbacks, and, by the van Kampen property, the front faces are also pullbacks. Moreover, by Lemma B.3, the front faces have the pushout complement property, and, by Lemma B.4, this also holds for \((16_0)\) and \((16_1)\) as compositions.

Thus, for a given \(n\), \((16_n)\) is the required pullback \((14_1)\) and \((14_1)+(1_i)\) with the pushout complement property using \(\tilde{K}_n = \tilde{K}_s\) and \(\tilde{L}_n = \tilde{L}_s\).

Moreover, we have pushout complements \((17_1)\) and \((17_1)+(1'_i)\) for \(t_{i,L} \circ l_i\) as in

\[
\begin{align*}
p_0 : & \quad ac_0 \triangleright\triangleleft L_0 \leftarrow L_0 \rightarrow K_0 \\
& \quad s_{i,L} \leftarrow (1'_i) \leftarrow w_i \leftarrow (3_i) \leftarrow e_{i,i} \\
& \quad \tilde{p}^*_{i} : \quad ac_i \triangleright\triangleleft L_i \leftarrow L_0 \\
& \quad t_{i,L} \leftarrow (17_i) \leftarrow \tilde{i}_i \leftarrow (18_i) \leftarrow \tilde{k}_i
\end{align*}
\]

Since \(ac_0\) and \(ac_i\) are complement-compatible for all \(i\), we have

\[
ac_i \cong \text{Shift}(s_{i,L}, ac_0) \wedge L \left( p_{i}^*, \text{Shift}(v_i, ac'_i) \right).
\]

For any \(ac'_i\), we have

\[
\begin{align*}
\text{Shift}(t_{i,L}, L(p_{i}^*, \text{Shift}(v_i, ac'_i)))) & \cong L \left( \tilde{p}^*_{i}, \text{Shift}(\tilde{k}_i \circ v_i, ac'_i) \right) \\
& \cong L \left( \tilde{p}^*_{i}, \text{Shift}(\tilde{v}, \text{Shift}(\tilde{l}_i, ac'_i)) \right)
\end{align*}
\]

since all squares are pushouts by pushout–pullback decomposition and the uniqueness of pushout complements. We define

\[
ac_i^* := \text{Shift}(\tilde{l}_i, ac'_i)
\]
as an application condition on \( L_0 \). It then follows that
\[
\hat{a}_c = \bigwedge_{i=1}^{n} \text{Shift}(t_{i,L}, ac_i)
\]
\[
\cong \bigwedge_{i=1}^{n} (\text{Shift}(t_{i,L} \circ s_{i,L}, ac_0) \land \text{Shift}(t_{i,L}, L(p_i^*, \text{Shift}(v_i, ac_i'))))
\]
\[
\cong \text{Shift}(t_{0,L}, ac_0) \land \bigwedge_{i=1}^{n} L(p_i^*, \text{Shift}(\tilde{v}, ac_i'^*)).
\]

For \( i = 0 \), we define
\[
ac'_0 = \bigwedge_{j=1}^{n} ac_j^*,
\]
so
\[
\hat{a}_c = \text{Shift}(t_{0,L}, ac_0) \land L(p_i^*, \text{Shift}(\tilde{v}, ac_0'))
\]
implies the complement-compatibility of \( ac_0 \) and \( \hat{a}_c \).

For \( i > 0 \), we have
\[
\text{Shift}(t_{0,L}, ac_0) \land L(p_i^*, \text{Shift}(\tilde{v}, ac_i')) \cong \text{Shift}(t_{i,L}, ac_i).
\]

We define
\[
ac'_i = \bigwedge_{j=1}^{n} ac_j^*,
\]
so
\[
\hat{a}_c = \text{Shift}(t_{i,L}, ac_i) \land L(p_i^*, \text{Shift}(\tilde{v}, ac_i'))
\]
implies the complement-compatibility of \( ac_i \) and \( \hat{a}_c \).

A.3. Proof of Fact 5.8

Proof. From Fact 4.8, each single direct transformation \( G \xrightarrow{p_i, m_i} G_i \) can be decomposed into a transformation
\[
G \xrightarrow{p_0, m_0} G_0 \xrightarrow{p_i, m_i} G_i
\]
with
\[
m_i^0 = m_i \circ s_{i,L},
\]
and since the bundle is \( s \)-amalgamable,
\[
m_0 = m_i \circ s_{i,L} = m_i^0
\]
and \( G_0 := G_0^i \) for all \( i = 1, \ldots, n \).

We now have to show the pairwise parallel independence.
From the constructions of the complement rule and the Concurrency Theorem, we obtain the following diagram for all $i = 1, \ldots, n$:

For $i \neq j$, the weakly independent matches mean we have a morphism $p_{ij} : L_{i0} \rightarrow D_j$ with

$$f_j \circ p_{ij} = m_i \circ u_i.$$

It follows that

$$f_j \circ p_{ij} \circ w_i = m_i \circ u_i \circ w_i$$

$$= m_i \circ s_{iL} \circ l_0$$

$$= m_0 \circ l_0$$

$$= m_j \circ s_{jL} \circ l_0$$

$$= m_j \circ u_j \circ w_j$$

$$= m_j \circ u_j \circ l_{j0} \circ s_{jK}$$

$$= m_j \circ l_j \circ s_{jK}$$

$$= f_j \circ k_j \circ s_{jK},$$

and with $f_j \in \mathcal{M}$, we have

$$p_{ij} \circ w_i = k_j \circ s_{jk}. \quad (*)$$

Now consider the pushout $(19_i) = (6_i) + (8_i)$ in comparison with object $D_j$ and morphisms $d_j \circ p_{ij}$ and $x_j \circ u_{j2} \circ s_{t3}$ as shown below:
Multi-amalgamation of rules with application conditions in $\mathcal{M}$-adhesive categories

We have

\[
d_j \circ p_{ij} \circ l_{i0} \circ s_{iK} = d_j \circ p_{ij} \circ w_i
\]
\[
= d_j \circ k_j \circ s_{jK} \quad \text{(by $\ast$)}
\]
\[
= x_j \circ r_{j0} \circ s_{jK}
\]
\[
= x_j \circ w_j \circ v_{j1} \circ s_{jK}
\]
\[
= x_j \circ u_{j2} \circ s_{j3} \circ s_{j1}
\]
\[
= x_j \circ u_{j2} \circ r_0
\]
\[
= x_j \circ u_{j2} \circ s_{i3} \circ s_{i1}.
\]

Now, pushout (19) induces a unique morphism $q_{ij}$ with

\[
q_{ij} \circ u_{i1} = d_j \circ p_{ij}
\]
\[
q_{ij} \circ l_i \circ s_{i4} = x_j \circ u_{j2} \circ s_{i3}.
\]

For the parallel independence of

\[
G_0 \xrightarrow{p_j, m_j} G_i
\]
\[
G_0 \xrightarrow{p_j, m_j} G_j,
\]

we have to show that $q_{ij} : \overline{L_i} \rightarrow \overline{D_j}$ satisfies

\[
f_j \circ q_{ij} = k_{i0} \circ e_{i2} =: \overline{m_j}.
\]

With $f_0 \in \mathcal{M}$ and

\[
f_0 \circ d_{j0} \circ p_{ij} = f_j \circ p_{ij}
\]
\[
= m_i \circ u_i
\]
\[
= f_0 \circ x_{i0}
\]

it follows that

\[
d_{j0} \circ p_{ij} = x_{i0}. \quad (\ast\ast)
\]

This means that

\[
f_j \circ q_{ij} \circ u_{i1} = f_j \circ d_j \circ p_{ij}
\]
\[
= g_0 \circ d_0 \circ p_{ij} \quad \text{(by $\ast\ast$)}
\]
\[
= g_0 \circ x_{i0}
\]
\[
= k_{i0} \circ e_{i2} \circ u_{i1}.
\]

We also have

\[
f_j \circ q_{ij} \circ l_i \circ s_{i4} = f_j \circ x_j \circ u_{j2} \circ s_{i3}
\]
\[
= k_{j0} \circ w_j \circ u_{j2} \circ s_{i3}
\]
\[
= k_{j0} \circ w_i \circ u_{j2} \circ s_{i3}
\]
\[
= k_{j0} \circ e_{i2} \circ \overline{l_i} \circ s_{i4}.
\]
Since (19) is a pushout, \( u_{i1} \) and \( \bar{t}_i \circ s_{i4} \) are jointly epimorphic, so
\[
\bar{f}_j \circ q_{ij} = k_{i0} \circ e_{i2}.
\]

If \( ac_0 \) and \( ac_i \) are not complement-compatible, then \( \bar{ac}_i = \text{true} \) and, trivially,
\[
\overline{g}_j \circ q_{ij} \models ac_i
\]
for all \( j \neq i \). Otherwise, we have
\[
g_j \circ p_{ij} \models ac'_i,
\]
and with
\[
g_j \circ p_{ij} = \overline{g}_j \circ d_j \circ p_{ij} = \overline{g}_j \circ q_{ij} \circ u_{i1}
\]
it follows that
\[
\overline{g}_j \circ q_{ij} \circ u_{i1} \models ac'_i,
\]
which is equivalent to
\[
\overline{g}_j \circ q_{ij} \models \text{Shift}(u_{i1}, ac'_i) = \bar{ac}_i.
\]

A.4. Proof of Theorem 5.9

Proof.

(1) Synthesis:
We have to show that \( \tilde{p}_s \) is applicable to \( G \) leading to an amalgamated transformation
\[
G \xrightarrow{\tilde{p}_s, \tilde{n}_s} H
\]
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 5.2.

Then we can apply Fact 4.8, which implies the decomposition of \( G \xrightarrow{\tilde{p}_s, \tilde{n}_s} H \) into
\[
G \xrightarrow{\tilde{p}_i, \tilde{m}_i} G_i \xrightarrow{q_i} H,
\]
where \( q_i \) 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 the pullbacks \((1), (2), (14), (15)\) (see Definition 5.1)
\[
\begin{array}{cccccc}
ac_0 & L_0 & l_0 & K_0 & r_0 & R_0 \\
s_{i,L} & (1) & s_{i,K} & (2) & s_{i,R} & s_{i,R} \\
ac_i & L_i & l_i & K_i & r_i & R_i \\
t_{i,L} & (14) & t_{i,K} & (15) & t_{i,R} & t_{i,R} \\
\bar{a}c_s & \bar{L}_s & l_s & \bar{K}_s & r_s & \bar{R}_s \\
\end{array}
\]
and the pushouts \((20), (21)\) (see proof of Fact 6.5) on the right
\[
\begin{array}{cccccc}
L_i & l_i & K_i & r_i & R_i \\
m_i & (20) & k_i & (21) & n_i \\
G & f_s & D_i & g_i & G_i \\
\end{array}
\]
Using Fact 5.8, we know that we can apply $p_0 \ via \ m_0$ to give a direct transformation $G \xrightarrow{p_0,m_0} G_0$ given by the pushouts (20) and (21):

Moreover, we can find decompositions of pushouts (20) and (20_i) into pushouts (1_i') and (22_i) and (22_i) and (23_i), respectively, by $\mathcal{M}$-pushout–pullback decomposition and the uniqueness of pushout complements as in

Since we have consistent matches, 

\[ m_i \circ s_{i,L} = m_0 \] 

for all $i = 1, \ldots, n$. The colimit $\bar{L}_s$ then implies that there is a unique morphism $\bar{m} : \bar{L}_s \to G$ with

\[ \bar{m} \circ t_{i,L} = m_i \]

\[ \bar{m} \circ t_{0,L} = m_0. \]

Moreover,

\[ m_i \models ac_i \Rightarrow \bar{m} \circ t_{i,L} \models ac_i \]

\[ \Rightarrow \bar{m} \models \text{Shift}(t_{i,L}, ac_i) \]

for all $i = 1, \ldots, n$, so

\[ \bar{m} \models \bigwedge_{i=1}^n \text{Shift}(t_{i,L}, ac_i). \]

The fact that we have weakly independent matches means that there exist morphisms $p_{ij}$ with

\[ f_j \circ p_{ij} = m_i \circ u_i \]
for $i \neq j$. We now construct $D$ as the limit of $(d_{i0})_{i=1,...,n}$ with morphisms $d_i$. Now $f_0$ being a monomorphism with

$$f_0 \circ d_{i0} \circ p_{ji} = f_i \circ p_{ji} = m_j \circ u_j = f_0 \circ x_{j0}$$

implies that

$$d_{i0} \circ p_{ji} = x_{j0},$$

so

$$d_{i0} \circ p_{ji} \circ l_{j0} = x_{j0} \circ l_{j0},$$

and, together with

$$d_{i0} \circ k_i = x_{i0} \circ l_{i0},$$

limit $D$ then implies that there exists a unique morphism $r_j$ with

$$d_i \circ r_j = p_{ji} \circ l_{j0}$$
$$d_i \circ r_i = k_i$$
$$d_0 \circ r_j = x_{j0} \circ l_{j0}.$$

Similarly, $f_j$ being a monomorphism with

$$f_j \circ p_{ij} \circ l_{i0} \circ s_{i,K} = m_i \circ u_i \circ w_i = m_i \circ s_{i,L} \circ l_0 = m_0 \circ l_0 = m_j \circ s_{j,L} \circ l_0 = m_j \circ l_j \circ s_{j,K} = f_j \circ k_j \circ s_{j,K}$$

implies that

$$p_{ij} \circ l_{i0} \circ s_{i,K} = k_j \circ s_{j,K}.$$

Now, colimit $\tilde{K}_s$ implies that there is a unique morphisms $\tilde{r}_j$ with

$$\tilde{r}_j \circ t_{i,K} = p_{ij} \circ l_{i0}$$
$$\tilde{r}_j \circ t_{j,K} = k_j$$
$$\tilde{r}_j \circ t_{0,K} = k_j \circ s_{j,K}.$$
Since

\[ d_{i0} \circ \tilde{r}_i \circ t_{i,K} = d_{i0} \circ k_i \]
\[ = q_i \circ l_0 \]
\[ = d_{j0} \circ p_{ij} \circ l_0 \]
\[ = d_{j0} \circ \tilde{r}_j \circ t_{i,K} \]

and

\[ d_{i0} \circ \tilde{r}_i \circ t_{0,K} = d_{i0} \circ k_i \circ s_{i,K} \]
\[ = k_0 \]
\[ = d_{j0} \circ \tilde{r}_j \circ t_{0,K} , \]

colimit $\tilde{K}_s$ implies that for all $i, j$ we have

\[ d_{i0} \circ \tilde{r}_i = d_{j0} \circ \tilde{r}_j =: \tilde{r} . \]

From limit $D$, it now follows that there exists a unique morphism $\tilde{k}$ with

\[ d_i \circ \tilde{k} = \tilde{r}_i \]
\[ d_0 \circ \tilde{k} = \tilde{r} . \]

We now have to show that (20$^s$) in

\[ f = f_0 \circ d_0 \] is a pushout.
With
\[
\begin{align*}
f \circ \tilde{k} \circ t_{i,K} &= f_0 \circ d_0 \circ \tilde{k} \circ t_{i,K} \\
&= f_0 \circ \tilde{r} \circ t_{i,K} \\
&= f_0 \circ d_{i0} \circ \tilde{r}_i \circ t_{i,K} \\
&= f_0 \circ d_{i0} \circ k_i \\
&= f_i \circ k_i \\
&= m_i \circ l_i \\
&= \tilde{m} \circ t_{i,L} \circ l_i \\
&= \tilde{m} \circ \tilde{l}_s \circ t_{i,K}
\end{align*}
\]
and
\[
\begin{align*}
f \circ \tilde{k} \circ t_{0,K} &= f_0 \circ d_0 \circ \tilde{k} \circ t_{0,K} \\
&= f_0 \circ \tilde{r} \circ t_{0,K} \\
&= f_0 \circ d_{i0} \circ \tilde{r}_i \circ t_{0,K} \\
&= f_0 \circ d_{i0} \circ k_i \circ s_{i,K} \\
&= f_0 \circ k_0 = m_0 \circ l_0 \\
&= \tilde{m} \circ t_{0,L} \circ l_0 \\
&= \tilde{m} \circ \tilde{l}_s \circ t_{0,K}
\end{align*}
\]
and \( \tilde{K}_s \) being a colimit, it follows that
\[
f \circ \tilde{k} = \tilde{m} \circ \tilde{l}_s,
\]
so the square commutes.

Pushout \((23_i)\) can be decomposed into pushouts \((24_i)\) and \((25_i)\) in

\[
\begin{array}{ccc}
K_i & \xrightarrow{r_i} & D \\
\downarrow t_{i0} & & \downarrow d_i \\
L_{i0} & \xrightarrow{x_{i0}} & P_i \\
\downarrow x_{i0} & & \downarrow y_{i0} \\
& & D_0
\end{array}
\]

Using Lemma B.5, it follows that \(D_0\) is the colimit of \((x_i)_{i=1,\ldots,n}\), because \((23_i)\) is a pushout, \(D\) is the limit of \((d_{i0})_{i=1,\ldots,n}\), and we have morphisms \(p_{ij}\) with \(d_{j0} \circ p_{ij} = q_i\). Lemma B.6 then implies that \((25)\) in

\[
\begin{array}{ccc}
+K_i & \xrightarrow{+t_{i0}} & +L_{i0} \\
\downarrow r & & \downarrow \pi \\
D & \xrightarrow{d_{i0}} & D_0
\end{array}
\]
is also a pushout, where \(+\) represents the coproduct construction with index \(i = 1,\ldots,n\) with injections \(t_{K_i}\) and \(t_{L_{i0}}\), respectively.
Consider the n-ary coequalisers:

- $\tilde{K}_s$ of $(t_{K_i} \circ s_{i,K} : K_0 \to +K_i)_{i=1,...,n}$
  (which is actually $\tilde{K}_s$ by construction of colimits);

- $\tilde{L}_0$ of $(\text{io}ta_{L_i} \circ w_i : K_0 \to +L_{i0})_{i=1,...,n}$
  (as already constructed in Fact 5.2);

- $D$ of $(\tilde{k} \circ t_{0,K} : K_0 \to D)_{i=1,...,n}$;

- $D_0$ of $(k_0 : K_0 \to D_0)_{i=1,...,n}$.

In the cube

the top square with identical morphisms is a pushout, the top cube commutes and the middle square is pushout (25) from above. Using Lemma B.7, it follows that the bottom square

constructed of the four coequalisers is a pushout too.

Now consider the cube
where the top and middle squares are pushouts and the two top cubes commute. Using Lemma B.7 again, it follows that \((20_s)\) in the bottom is actually a pushout, where

\[(27) = (1'_i) + (17_i)\]

is a pushout by composition:

\[
\begin{array}{ccc}
K_0 & \xrightarrow{t_0} & \tilde{L}_0 \\
\downarrow & & \downarrow \\
L_0 & \xrightarrow{t_{0,K}} & \tilde{L}_s
\end{array}
\]

We can now construct pushout \((21_s)\), which completes the direct transformation

\[G \xrightarrow{p_i, m_i} H.\]

\[
\begin{array}{cccc}
\tilde{L}_s & \xrightarrow{i_s} & \tilde{K}_s & \xrightarrow{\tilde{r}_s} & \tilde{R}_s \\
\downarrow & & \downarrow & & \downarrow \\
G & \xrightarrow{f} & D & \xrightarrow{g} & H
\end{array}
\]

(2) **Analysis:**

Using the kernel morphisms \(t_i\), we obtain transformations

\[G \xrightarrow{p_i, m_i} G_i \xrightarrow{q_i} H\]

from Fact 4.8 with

\[m_i = \tilde{m} \circ t_{i,L}.\]

We have to show that this bundle of transformation is \(s\)-amalgamable. Applying Fact 4.8 again, we obtain transformations

\[G \xrightarrow{p_0, m_{0i}} G_0^i \xrightarrow{\overline{q}_i} G_i\]

with

\[m_0^i = m_i \circ s_{i,L}.\]

It follows that

\[m_0^i = m_i \circ s_{i,L} = \tilde{m} \circ t_{i,L} \circ s_{i,L} = \tilde{m} \circ t_{0,L} = \tilde{m} \circ t_{j,L} \circ s_{j,L} = m_j \circ s_{j,L},\]

so we have consistent matches with \(m_0 := m_0^i\) well defined and \(G_0 = G_0^i\).

We still need to show the weakly independent matches. Given the above transformations, we have pushouts \((20_0)\), \((20_i)\) and \((20_s)\) as above. We can find decompositions of \((20_0)\)
and (20s) into pushouts (27) + (28) and (26) + (28), respectively:

\[
\begin{array}{c}
K_0 \xrightarrow{l_0} L_0 \\
\tilde{K}_s \xrightarrow{\tilde{a}} L_s \\
D \xrightarrow{d_0} D_0 \\
\end{array}
\]

Using pushout (26) and Lemma B.8, it follows that (25) as above is a pushout since \(\tilde{K}_s\) is the colimit of \((s_i,L)_{i=1,\ldots,n}\) and \(L_0\) is the colimit of \((w_i)_{i=1,\ldots,n}\), and \(id_{K_0}\) is obviously an epimorphism.

Lemma B.6 now implies that there is a decomposition into pushouts (24i) with colimit \(D_0\) of \((x_i)_{i=1,\ldots,n}\) and pushout (25i) by the \(M\)-pushout–pullback decomposition:

\[
\begin{array}{c}
K_i \xrightarrow{r_i} D \xrightarrow{d_i} D_i \\
K_0 \xrightarrow{w_i} L_i \xrightarrow{x_i} P_i \xrightarrow{y_{i0}} D_i \\
L_0 \xrightarrow{s_i,L} L_i \xrightarrow{m_i} G \\
\end{array}
\]

Since \(D_0\) is the colimit of \((x_i)_{i=1,\ldots,n}\) and (25j) is a pushout, it follows that \(D_j\) is the colimit of \((x_i)_{i=1,\ldots,j-1,\ldots,j+1,\ldots,n}\) with morphisms \(q_{ij}: P_i \rightarrow D_j\) and \(d_{j0} \circ q_{ij} = y_{i0}\):

\[
\begin{array}{c}
P_i \xrightarrow{x_i} D_j \xrightarrow{d_{j0}} D_0 \\
\end{array}
\]

Hence, we obtain for all \(i \neq j\), a morphism

\[p_{ij} = q_{ij} \circ x_{i0}\]

and

\[f_j \circ p_{ij} = f_0 \circ d_{j0} \circ q_{ij} \circ x_{i0} = f_0 \circ y_{i0} \circ x_{i0} = m_i \circ u_i.\]
(3) **Bijective correspondence:**
Because of the uniqueness of the constructions used, 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 are used in the proofs of the main theorems: Lemmas B.1 and B.2 are used in the proof of Theorem 4.4; Lemmas B.3 and B.4 are used in the proof of Fact 5.2; and Lemmas B.5, B.6, B.7 and B.8 are used in the proof of Theorem 5.9.

**Lemma B.1 (M complement property).** If

\[
\begin{array}{ccc}
A & \xrightarrow{m} & B \\
\downarrow{f} & & \downarrow{g} \\
C & \xrightarrow{n} & D
\end{array}
\]

is a pushout and

\[
\begin{array}{ccc}
A & \xrightarrow{m} & B \\
\downarrow{f'} & & \downarrow{g} \\
C' & \xrightarrow{n'} & D
\end{array}
\]

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 $n \in \mathcal{M}$ also because (1) is a pushout. We construct the pullback

\[
\begin{array}{ccc}
A & \xrightarrow{f'} & C'' \\
\downarrow{f} & & \downarrow{v} \\
C' & \xrightarrow{n'} & D
\end{array}
\]

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

\[v \circ f^* = f',
\]

\[v' \circ f^* = f.
\]
Now consider the cube

\[
\begin{array}{c}
A & \xrightarrow{f^*} & C'' \\
\downarrow{m} & & \downarrow{v} \\
\downarrow{f'} & & \\
B & \xrightarrow{g} & D
\end{array}
\]

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, the back right face is a pullback too, and the VK property then implies that the top face is a pushout. Since

\[
\begin{array}{c}
A & \xrightarrow{f} & C'' \\
\downarrow{id_A} & & \downarrow{v} \\
\downarrow{id_A} & & \\
A & \xrightarrow{f'} & C'
\end{array}
\]

is a pushout, and pushout objects are unique up to isomorphism, this implies that \(v\) is an isomorphism and \(C'' \cong C'\). We now define \(c := v' \circ v^{-1}\) and have

\[
c \circ f' = v' \circ v^{-1} \circ f' = v' \circ f^* = f
\]

and

\[
n \circ c = n \circ v' \circ v^{-1} = n',
\]

and \(c \in \mathcal{M}\) by decomposition of \(\mathcal{M}\)-morphisms.

**Lemma B.2 (\(\mathcal{M}\) pullback-pushout decomposition).** Consider

\[
\begin{array}{c}
A & \xrightarrow{f} & B & \xrightarrow{g} & E \\
\downarrow{m} & & \downarrow{n} & \downarrow{o} & \\
C & \xrightarrow{f'} & D & \xrightarrow{g'} & F
\end{array}
\]

If (1) + (2) is a pullback, (1) is a pushout, (2) commutes and \(o \in \mathcal{M}\), then (2) is a pullback too.
Proof. With \( o \in \mathcal{M} \) and the fact that (1) + (2) is a pullback and (1) is a pushout, we have that \( m, n \in \mathcal{M} \). We construct the pullback

\[
\begin{array}{ccc}
B & \xrightarrow{\bar{g}} & E \\
\downarrow{\pi} & & \downarrow{o} \\
D & \xrightarrow{g'} & F
\end{array}
\]

of \( o \) and \( g' \). It then follows that \( \pi \in \mathcal{M} \) and we get an induced morphism \( b : B \to \bar{B} \) with

\[
\bar{g} \circ b = g, \\
\pi \circ b = n,
\]

and, by decomposition of \( \mathcal{M} \)-morphisms, \( b \in \mathcal{M} \).

By pullback decomposition, (4) is a pullback too:

\[
\begin{array}{cc}
A & \xrightarrow{f} & B \\
\downarrow{m} & & \downarrow{b} \\
C & \xrightarrow{f'} & D
\end{array}
\]

\[
\begin{array}{cc}
\bar{B} & \xrightarrow{\bar{g}} & E \\
\downarrow{\pi} & & \downarrow{o} \\
D & \xrightarrow{g'} & F
\end{array}
\]

So we can apply Lemma B.1 with pushout (1) and \( \bar{\pi} \in \mathcal{M} \) to obtain a unique morphism \( \bar{b} \in \mathcal{M} \) with \( n \circ \bar{b} = \bar{\pi} \) and \( \bar{b} \circ b \circ f = f \):

\[
\begin{array}{cc}
A & \xrightarrow{m} & C \\
\downarrow{f} & & \downarrow{f'} \\
B & \xrightarrow{\bar{b} \circ f} & D \\
\downarrow{\pi} & & \downarrow{\bar{\pi}} \\
B & \xrightarrow{n} & D
\end{array}
\]

Now \( n \in \mathcal{M} \) and

\[
n \circ \bar{b} \circ b = \bar{\pi} \circ b = n
\]

implies that

\[
\bar{b} \circ b = id_B,
\]

and, similarly, \( \pi \in \mathcal{M} \) and

\[
\bar{\pi} \circ b \circ \bar{b} = n \circ \bar{b} = \bar{\pi}
\]

implies that \( b \circ \bar{b} = id_{\bar{B}} \), which means that \( B \) and \( \bar{B} \) are isomorphic, so (2) is a pullback too. \( \square \)
Lemma B.3. Given the commutative cube

\[
\begin{array}{cccc}
C' & A' & B' \\
\downarrow c & \downarrow a & \downarrow b \\
C & A & B \\
\downarrow f & \downarrow m & \downarrow g \\
D & D' & B' \\
\downarrow d & \downarrow b & \downarrow g' \\
D & D' & B' \\
\end{array}
\]

with the bottom face a pushout, 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. We construct the initial pushout (1) over \( f \):

\[
\begin{array}{cccc}
C' & A' & B' \\
\downarrow c & \downarrow a & \downarrow b \\
C & A & B \\
\downarrow f & \downarrow m & \downarrow g \\
D & D' & B' \\
\downarrow d & \downarrow f & \downarrow g' \\
D & D' & B' \\
\end{array}
\]

Since the back left face has a pushout complement, there is a morphism \( b^* : B_f \rightarrow A' \) such that \( a \circ b^* = b_f \). Since the bottom face is a pushout, the square

\[
\begin{array}{ccc}
B_f & \rightarrow & B \\
\downarrow \alpha_f & & \downarrow g \\
C_f & \rightarrow & D \\
\end{array}
\]

as the composition, is the initial pushout over \( g \). Now

\[
b \circ m' \circ b^* = m \circ a \circ b^* = m \circ b_f,
\]

so the pushout complement of \( g \circ b \) exists. \( \square \)

Lemma B.4. If we are given pullbacks (1) and (2)
with pushout complements over $f' \circ m$ and $g' \circ n$, respectively, then $(1) + (2)$ also 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 Lemma B.1, there are morphisms $c$ and $e$ such that

\begin{align*}
c \circ f &= f^* \\
n^* \circ c &= n \\
e \circ g &= g^* \\
o^* \circ e &= o.
\end{align*}

Now $(2')$ can be decomposed into pushouts $(3)$ and $(4)$:

\[
\begin{array}{c}
\text{(1)} \\
\text{(3)} \\
\text{(4)}
\end{array}
\]

and $(1') + (4)$ is also a pushout and the pushout complement of $(g' \circ f') \circ m$.

**Lemma B.5.** If we are given:

— the pushouts

\[
\begin{array}{c}
\text{(1)}_i \\
\text{(3)}_i
\end{array}
\]

and

\[
\begin{array}{c}
\text{(3)}_i \\
\text{(2)}_i
\end{array}
\]

with $b_i \in M$ for $i = 1, \ldots, n$;
morphisms $f_{ij} : B_i \to C_j$ with $c_j \circ f_{ij} = d_i$ for all $i \neq j$:

$$
\begin{array}{ccc}
B_i & \xrightarrow{f_{ij}} & C_j \\
\downarrow{d_i} & & \downarrow{c_j} \\
D & &
\end{array}
$$

the limit

$$
\begin{array}{ccc}
C_j & \xrightarrow{e_j} & D \\
\downarrow{(2)} & & \downarrow{e_j} \\
E & \xrightarrow{\pi} &
\end{array}
$$

of $(c_j)_{j=1,\ldots,n}$ such that $g_i$ is the induced morphism into $E$ with

$$
\begin{aligned}
e_i \circ g_i &= a_i \\
e_j \circ g_i &= f_{ij} \circ b_i
\end{aligned}
$$

using

$$
c_j \circ f_{ij} \circ b_i = d_i \circ b_i = c_i \circ a_i,
$$

then we have

$$
\begin{array}{ccc}
E & \xrightarrow{h_i} & F_i \\
\downarrow{\pi} & & \downarrow{l_i} \\
D & &
\end{array}
$$

is the colimit of $(h_i)_{i=1,\ldots,n}$, where $l_i$ is the induced morphism from pushout $(3_i)$ compared with

$$
\overline{\pi} \circ g_i = c_i \circ e_i \circ g_i = c_i \circ a_i = d_i \circ b_i.
$$

Proof. We use induction over $n$:

Base case $(n = 1)$:
For $n = 1$, we have that $C_1$ is the limit of $c_1$, that is, $E = C_1$. It follows that $F_1 = C_1$ for the pushout $(3_1) = (1_1)$, so we have

\[
\begin{array}{c}
A_1 \xrightarrow{a_1} C_1 \\
\downarrow{b_1} & \downarrow{c_i} \\
B_1 \xrightarrow{d_i} D
\end{array}
\]

and

\[
\begin{array}{c}
C_1 \xrightarrow{C_1} D \\
\downarrow{C_1} \quad \quad \downarrow{C_1} \\
(2) \quad \quad (4_1)
\end{array}
\]

and it is obvious that

is a colimit.

— **Induction step** $(n \to n + 1)$:

Consider:

- the pushouts

\[
\begin{array}{c}
A_i \xrightarrow{a_i} C_i \\
\downarrow{b_i} & \downarrow{c_i} \\
B_i \xrightarrow{d_i} D
\end{array}
\]

with $b_i \in \mathcal{M}$ for $i = 1, \ldots, n + 1$;

- the morphisms $f_{ij} : B_i \to C_j$ with $c_j \circ f_{ij} = d_i$ for all $i \neq j$; and

- the limits

\[
\begin{array}{c}
C_i \xrightarrow{C_i} D \\
\downarrow{C_i} \quad \quad \downarrow{C_i} \\
(2_n) \quad \quad (2_{n+1})
\end{array}
\]

of $(c_i)_{i=1,\ldots,n}$ and $(c_i)_{i=1,\ldots,n+1}$, respectively, leading to the pullback

\[
\begin{array}{c}
E_{n+1} \xrightarrow{e_{n+1}} C_{n+1} \\
\downarrow{p_{n+1}} & \downarrow{c_{n+1}} \\
E_n \xrightarrow{e_{n+1}} D
\end{array}
\]

by the construction of limits.
Moreover, \( g_{in} \) and \( g_{in+1} \) are the induced morphisms into \( E_n \) and \( E_{n+1} \), respectively, leading to the pushouts

\[
\begin{array}{c}
A_i \\ b_i
\end{array}
\rightarrow
\begin{array}{c}
E_n \\ h_{in}
\end{array}
\leftarrow
\begin{array}{c}
B_i \\ k_{in}
\end{array}
\]

and

\[
\begin{array}{c}
A_i \\ g_{in+1}
\end{array}
\rightarrow
\begin{array}{c}
E_{n+1} \\ h_{in+1}
\end{array}
\leftarrow
\begin{array}{c}
B_i \\ k_{in+1}
\end{array}
\]

By the induction hypothesis,

\[
\begin{array}{c}
E_n \\ h_{in}
\end{array}
\rightarrow
\begin{array}{c}
F_{in}
\end{array}
\leftarrow
\begin{array}{c}
D
\end{array}
\]

is the colimit of \( (h_{in})_{i=1,...,n} \), and we have to show that

\[
\begin{array}{c}
E_{n+1} \\ h_{in+1}
\end{array}
\rightarrow
\begin{array}{c}
F_{in+1}
\end{array}
\leftarrow
\begin{array}{c}
D
\end{array}
\]

is the colimit of \( (h_{in+1})_{i=1,...,n+1} \).

Since \((2_n)\) is a limit and

\[c_i \circ f_{n+1i} = d_{n+1}\]

for all \(i = 1, \ldots, n\), we obtain a unique morphism \(m_{n+1}\) with

\[e_{in} \circ m_{n+1} = f_{n+1i}\]
\[\epsilon_{n} \circ m_{n+1} = d_{n+1}.
\]

Since \((1_{n+1})\) is a pushout and \((5_{n+1})\) is a pullback, by \(\mathcal{M}\)-pushout–pullback decomposition, \((5_{n+1})\) and \((6_{n+1})\) are pushouts too:

\[
\begin{array}{c}
A_{n+1} \\ b_{n+1}
\end{array}
\rightarrow
\begin{array}{c}
E_{n+1} \\ \epsilon_{n+1}
\end{array}
\leftarrow
\begin{array}{c}
C_{n+1}
\end{array}
\]

\[
\begin{array}{c}
B_{n+1} \\ m_{n+1}
\end{array}
\rightarrow
\begin{array}{c}
E_{n} \\ \epsilon_{n}
\end{array}
\leftarrow
\begin{array}{c}
D
\end{array}
\]

\[
\begin{array}{c}
A_{n+1} \\ g_{n+1n+1}
\end{array}
\rightarrow
\begin{array}{c}
E_{n+1} \\ \epsilon_{n+1n+1}
\end{array}
\leftarrow
\begin{array}{c}
C_{n+1} \\ \epsilon_{n+1}\end{array}
\]

\[
\begin{array}{c}
B_{n+1} \\ m_{n+1}
\end{array}
\rightarrow
\begin{array}{c}
E_{n} \\ \epsilon_{n}
\end{array}
\leftarrow
\begin{array}{c}
D
\end{array}
\]

\[
\begin{array}{c}
A_{n+1} \\ g_{n+1n+1}
\end{array}
\rightarrow
\begin{array}{c}
E_{n+1} \\ \epsilon_{n+1n+1}
\end{array}
\leftarrow
\begin{array}{c}
C_{n+1} \\ \epsilon_{n+1}\end{array}
\]
So \( F_{n+1} = E_n \). From pushout (3\(_{n+1}\)) and

\[
    h_{in} \circ p_{n+1} \circ g_{in+1} = h_{in} \circ g_{in} = k_{in} \circ b_i
\]

we get an induced morphism \( q_{in+1} \) with

\[
    q_{in+1} \circ h_{in+1} = h_{in} \circ p_{n+1}
\]

\[
    q_{in+1} \circ k_{in+1} = k_{in},
\]

and from pushout decomposition, (7\(_{in+1}\)) is a pushout too:

\[
\begin{array}{cccc}
A_{i} & \xrightarrow{g_{in}} & E_{n+1} & \xrightarrow{p_{n+1}} E_{n2} \\
B_{i} & \xrightarrow{k_{in+1}} & F_{in+1} & \xrightarrow{q_{in+1}} F_{in} \\
& \xrightarrow{h_{in+1}} & (3_{in+1}) & \xrightarrow{(7_{in+1})} (7_{in+1}) \\
& \xrightarrow{h_{in}} & & \xrightarrow{k_{in}} \\
\end{array}
\]

To show that (4\(_{n+1}\)) is a colimit, consider an object \( X \) and morphisms \((x_i)\) and \( y \) with

\[
    x_i \circ h_{in+1} = y
\]

for \( i = 1, \ldots, n \) and

\[
    x_{n+1} \circ p_{n+1} = y.
\]

From pushout (7\(_{in+1}\)), we obtain a unique morphism \( z_i \) with

\[
    z_i \circ q_{in+1} = x_i
\]

\[
    z_i \circ h_{in} = x_{n+1}.
\]

Now, colimit (4\(_n\)) induces a unique morphism \( z \) with

\[
    z \circ \overline{\pi_n} = x_{n+1}
\]

\[
    z \circ l_{in} = z_i.
\]
It then follows directly that
\[
z \circ l_{in+1} = z \circ l_{in} \circ q_{in+1} \\
= z_i \circ q_{in+1} \\
= x_i
\]
and
\[
z \circ e_{n+1} = z \circ e_n \circ p_{n+1} \\
= x_{n+1} \circ p_{n+1} \\
= y.
\]

The uniqueness of $z$ then follows directly from the construction, so $(4_{n+1})$ is the required colimit.

\[\square\]

**Lemma B.6.** Given the diagrams
\[
\begin{array}{c}
A_i \xrightarrow{a_i} C \\
\downarrow b_i & (1_i) & \downarrow c_i \\
B_i \xrightarrow{d_i} D_i
\end{array}
\]
for $i = 1, \ldots, n$,

\[
\begin{array}{c}
C \xrightarrow{c_i} D_i \\
\downarrow c & (2) & \downarrow c_i \\
E
\end{array}
\]
and
\[
\begin{array}{c}
+A_i \xrightarrow{b} +B_i \\
\downarrow a & (3) & \downarrow e \\
C \xrightarrow{c} E
\end{array}
\]
with $b = +b_i$, and $a$ and $e$ induced by the coproducts $+A_i$ and $+B_i$, respectively, we have:

(a) If $(1_i)$ is a pushout and $(2)$ a colimit, then $(3)$ is also a pushout.
(b) If (3) is a pushout, then there is a decomposition into pushout (1) and colimit (2) with \( e_i \circ d_i = e \circ i_B \):

\[
\begin{array}{c}
A_i \xrightarrow{a_i} C \\
\downarrow_i \quad \downarrow_i \\
A_i \xrightarrow{b_i} B_i \\
\downarrow_i \quad \downarrow_i \\
E
\end{array}
\]

Proof.

(a) We assume we are given an object \( X \) and morphisms \( y, z \) with \( y \circ a = z \circ b \):

\[
\begin{array}{c}
+ A_i \xrightarrow{b} + B_i \\
\downarrow a \\
C \\
\downarrow c \\
E \xrightarrow{z} X
\end{array}
\]

From pushout (1), we obtain with

\[
z \circ i_B \circ b_i = z \circ b \circ i_{A_i} = y \circ a \circ i_{A_i} = y \circ a_i,
\]

a unique morphism \( x_i \)

\[
\begin{array}{c}
A_i \xrightarrow{a_i} C \\
\downarrow_{b_i} \quad \downarrow_{c_i} \\
B_i \xrightarrow{d_i} D_i \\
\downarrow_y \\
X \xrightarrow{z \circ i_B} X
\end{array}
\]

with

\[
x_i \circ c_i = y \\
x_i \circ d_i = z \circ i_B.
\]

Now colimit (2) implies a unique morphism \( x \) with

\[
x \circ c = y \\
x \circ e_i = x_i.
\]
Multi-amalgamation of rules with application conditions in \( \mathcal{M} \)-adhesive categories

It then follows that

\[
x \circ e \circ i_B = x \circ e_i \circ d_i = x_i \circ d_i = z \circ i_B,
\]

and since \( z \) is unique with respect to \( z \circ i_B \), it follows from the coproduct that \( z = x \circ e \).

The uniqueness of \( x \) follows from the uniqueness of \( x \) and \( x_i \), so (3) is a pushout.

(b) We define \( a_i := a \circ i_{A_i} \) and construct the pushout

With

\[
e \circ i_B \circ b_i = e \circ b \circ i_{A_i} = c \circ a_i,
\]

pushout (1) induces a unique morphism \( e_i \) with

\[
e_i \circ d_i = e \circ i_B, \\
e_i \circ c_i = c.
\]

Given an object \( X \) and morphisms \( y \) and \( y_i \) with \( y_i \circ c_i = y \),
we obtain a morphism $z$ with

$$z \circ i_{Bi} = y_i \circ d_i$$

from coproduct $+B_i$.

So we have

$$y \circ a \circ i_{Ai} = y_i \circ c_i \circ a_i$$
$$= y_i \circ d_i \circ b_i$$
$$= z \circ i_{Bi} \circ b_i$$
$$= z \circ b \circ i_{Ai},$$

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 $x \circ e = z$:

From pushout (1), and using

$$x \circ e_i \circ d_i = x \circ e \circ i_{Bi}$$
$$= z \circ i_{Bi}$$
$$= y_i \circ d_i$$

and

$$x \circ e_i \circ c_i = x \circ c$$
$$= y$$
$$= y_i \circ c_i,$$

it then follows that $x \circ e_i = y_i$, so (2) is a colimit.

\[\square\]

**Lemma B.7.** Consider the colimits

it follows that $x \circ e_i = y_i$, so (2) is a colimit.
such that
\[
\begin{array}{c}
A_i \\ A_j
\end{array}
\begin{array}{c}
\xrightarrow{f_i} \\
\xrightarrow{g_i}
\end{array}
\begin{array}{c}
B_i \\ B_j
\end{array}
\begin{array}{c}
(5_i) \\
(\alpha_k)
\end{array}
\begin{array}{c}
C_i \\ C_j
\end{array}
\begin{array}{c}
\xrightarrow{h_i} \\
\xrightarrow{h_j}
\end{array}
\begin{array}{c}
D_i \\ D_j
\end{array}
\]

is a pushout for all \(i = 1, \ldots, n\) and
\[
\begin{array}{c}
A_i \\ A_j
\end{array}
\begin{array}{c}
\xrightarrow{f_i} \\
\xrightarrow{g_i}
\end{array}
\begin{array}{c}
B_i \\ B_j
\end{array}
\begin{array}{c}
(6_k) \\
(\alpha_k)
\end{array}
\begin{array}{c}
C_i \\ C_j
\end{array}
\begin{array}{c}
\xrightarrow{h_i} \\
\xrightarrow{h_j}
\end{array}
\begin{array}{c}
D_i \\ D_j
\end{array}
\begin{array}{c}
(7_k) \\
(8_k)
\end{array}
\begin{array}{c}
A_i \\ A_j
\end{array}
\begin{array}{c}
\xrightarrow{a_k} \\
\xrightarrow{b_k}
\end{array}
\begin{array}{c}
B_i \\ B_j
\end{array}
\begin{array}{c}
(9_k) \\
d_k
\end{array}
\begin{array}{c}
C_i \\ C_j
\end{array}
\begin{array}{c}
\xrightarrow{c_k} \\
\xrightarrow{d_k}
\end{array}
\begin{array}{c}
D_i \\ D_j
\end{array}
\]

commute for all \(k = 1, \ldots, m\). Then
\[
\begin{array}{c}
\bar{A} \\ \bar{C}
\end{array}
\begin{array}{c}
\xrightarrow{\bar{f}} \\
\xrightarrow{\bar{g}}
\end{array}
\begin{array}{c}
\bar{B} \\ \bar{D}
\end{array}
\begin{array}{c}
\bar{h}
\end{array}
\begin{array}{c}
(10)
\end{array}
\]

is a pushout too.

**Proof.** The morphisms \(\bar{f}, \bar{g}, \bar{h}\) and \(\bar{k}\) are uniquely induced by the colimits. We will just show the case for the morphism \(\bar{f}\) as an example.

From colimit (1), with
\[
\bar{b}_j \circ f_j \circ a_k = \bar{b}_j \circ b_k \circ f_i = \bar{b}_i \circ f_i,
\]
we obtain a unique morphism \(\bar{f}\) with
\[
\bar{f} \circ \bar{a}_i = \bar{b}_i \circ f_i.
\]

It then follows directly that
\[
\bar{k} \circ \bar{h} = \bar{h} \circ \bar{f}.
\]

Now consider an object \(X\) and morphisms \(y\) and \(z\) with
\[
y \circ \bar{g} = z \circ \bar{f}.
\]
From pushout (5) with
\[ y \circ c_i \circ g_i = y \circ \overline{g} \circ \overline{a}_i = z \circ \overline{f} \circ \overline{a}_i = z \circ \overline{b}_i \circ f_i, \]
we obtain a unique morphism \( x_i \) with
\[ x_i \circ k_i = y \circ \overline{c}_i \]
\[ x_i \circ h_i = z \circ \overline{b}_i. \]

For all \( k = 1, \ldots, m \), we have
\[ x_j \circ d_k \circ k_i = x_j \circ k_j \circ c_k \]
\[ = y \circ \overline{c}_j \circ c_k \]
\[ = y \circ \overline{c}_i \]
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).

\[
x \circ \overline{h} \circ \overline{b}_i = x \circ \overline{d}_i \circ h_i \\
= x_i \circ h_i \\
= z \circ \overline{b}_i
\]

implies that \( x \circ \overline{h} = z \):

\[
\begin{array}{c}
B_i \\
\downarrow_{\overline{b}_i} \downarrow \downarrow_{\underline{b}_j} \\
B \\
\downarrow_{z \circ \overline{b}_i} \downarrow_{z \circ \overline{b}_j} \\
X
\end{array}
\]

Similarly,

\[
x \circ \overline{k} = y,
\]

and the uniqueness follows from the uniqueness of \( x \) with respect to (4), so (10) is indeed a pushout.

\[\square\]

**Lemma B.8.** We assume:

— colimits

\[
\begin{array}{c}
A \\
\downarrow_{\alpha} \\
A_i \\
\downarrow_a \\
A
\end{array}
\quad
\begin{array}{c}
B \\
\downarrow_{\beta} \\
B_i \\
\downarrow_b \\
B
\end{array}
\]

such that

\[
\begin{array}{c}
A \\
\downarrow_{a_i} \\
A_i \\
\downarrow_{f_i} \\
B_i
\end{array}
\]

commutes for all \( i = 1, \ldots, n \);

— \( f \) is an epimorphism; and

— the square

\[
\begin{array}{c}
A \\
\downarrow_{\tau} \\
C \\
\downarrow_e \\
D
\end{array}
\quad
\begin{array}{c}
B \\
\downarrow_{\tau} \\
D
\end{array}
\]

is a pushout with \( \overline{f} \) induced by colimit (1).
Then

\[ +A_i \xrightarrow{+f_i} +B_i \]

is a pushout also, 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

\[
\overline{f} \circ \overline{a}_i = \overline{b}_i \circ f_i \\
\overline{f} \circ \overline{a} = \overline{b} \circ f.
\]

From the coproducts, we obtain induced morphisms:

— \( c \) with \( c \circ i_{A_i} = \overline{c} \circ \overline{a}_i \)

\[
A_i \xrightarrow{i_{A_i}} +A_i \xrightarrow{c} C
\]

— \( d \) with \( d \circ i_{B_i} = \overline{d} \circ \overline{b}_i \)

\[
B_i \xrightarrow{i_{B_i}} +B_i \xrightarrow{d} D
\]

Moreover, for all \( i = 1, \ldots, n \), we have

\[
d \circ (+f_i) \circ i_{A_i} = d \circ i_{B_i} \circ f_i \\
= \overline{d} \circ \overline{b}_i \circ f_i \\
= \overline{d} \circ \overline{f} \circ \overline{a}_i \\
= e \circ \overline{c} \circ \overline{a}_i \\
= e \circ c \circ i_{A_i}.
\]
Uniqueness of the induced coproduct morphisms leads to
\[ d \circ (+f_i) = e \circ c, \]
that is, (5) commutes.

\[ A_i \xrightarrow{i_{A_i}} +A_i \]
\[ f_i \]
\[ B_i \xrightarrow{i_{B_i}} +B_i \]

We now have to show that (5) is a pushout:

\[ +A_i \xrightarrow{+f_i} +B_i \]
\[ c \]
\[ C \]
\[ x \]
\[ y \]
\[ D \]
\[ X \]

Given morphisms \( x \) and \( y \) with
\[ x \circ c = y \circ (+f_i), \]
we have
\[ y \circ i_{B_i} \circ b_i \circ f = y \circ i_{B_i} \circ f_i \circ a_i \]
\[ = y \circ (+f_i) \circ i_{A_i} \circ a_i \]
\[ = x \circ c \circ i_{A_i} \circ a_i \]
\[ = x \circ c \circ \overline{a}_i \circ a_i \]
\[ = x \circ \overline{c} \circ \overline{a} \]

for all \( i = 1, \ldots, n \). The fact that \( f \) is an epimorphism implies that
\[ y \circ i_{B_i} \circ b_i = y \circ i_{B_j} \circ b_j \]
for all \( i, j \). We now define \( y' := y \circ i_{B_i} \circ b_i \) and from colimit (2)

\[ B \]
\[ \overline{b}_i \]
\[ (2) \]
\[ B_i \]
\[ x \]
\[ y' \]
\[ y \circ i_{B_i} \]

Downloaded from https://www.cambridge.org/core. Universitaetsbibliothek, on 26 Oct 2017 at 13:50:14, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S0960129512000345
we obtain a unique morphism \( y \) with \( y \circ b_i = y \circ i_B \) and \( y \circ b = y' \):

Now

\[
x \circ c \circ i_A = x \circ c \circ i_A
\]
\[
= y \circ (f_i) \circ i_A
\]
\[
= y \circ i_B \circ f_i
\]
\[
= y \circ b_i \circ f_i
\]
\[
= y \circ f \circ a_i
\]

and

\[
x \circ c \circ a = x \circ c \circ a
\]
\[
= y \circ f \circ a
\]
\[
= y \circ f \circ a
\]

and the uniqueness of the induced colimit morphism implies that

\[
y \circ f = x \circ c.
\]

This means that \( X \) can be compared to pushout (4), and we obtain a unique morphism \( z \) with \( z \circ d = y \) and \( z \circ e = x \):

Now

\[
z \circ d \circ i_B = z \circ d \circ b_i
\]
\[
= y \circ b_i
\]
\[
= y \circ i_B,
\]

so \( z \circ d = y \). Similarly, the uniqueness of \( z \) with respect to the pushout property of (5) also follows, so (5) is a pushout. \( \square \)
Multi-amalgamation of rules with application conditions in $\mathcal{M}$-adhesive categories

References


