Under consideration for publication in Math. Struct. in Comp. Science

Multi-Amalgamation of Rules with Application Conditions in $\mathcal{M}$-Adhesive Categories

ULRIKE GOLAS[1], ANNEGRET HABEL[2] and HARTMUT EHRIG[3]
[1] *Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany, golas@zib.de*
[2] *Universität Oldenburg, Germany, annegret.habel@informatik.uni-oldenburg.de*
[3] *Technische Universität Berlin, Germany, ehrig@cs.tu-berlin.de*

Received September 2011

Amalgamation is a well-known concept for graph transformations in order to model synchronized parallelism of rules with shared subrules and corresponding transformations. This concept is especially important for an adequate formalization of the operational semantics of statecharts and other visual modeling languages, where typed attributed graphs are used for multiple rules with nested application conditions. However, the theory of amalgamation for the double-pushout approach has been developed up to now only on a set-theoretical basis for pairs of standard graph rules without any application conditions.
For this reason, we present the theory of amalgamation in this paper for $\mathcal{M}$-adhesive categories, a slightly more general framework than (weak) adhesive HLR categories, for a bundle of rules with (nested) application conditions. The two main results are the Complement Rule Theorem showing how to construct a minimal complement rule for each subrule and the Multi-Amalgamation Theorem, which generalizes the well-known Parallelism and Amalgamation Theorems to the case of multiple synchronized parallelism. For the application of the largest amalgamated rule we use maximal matchings, which are computed depending on the actual instance graph. The constructions are illustrated by a small but meaningful running example, while a more complex case study concerning the firing semantics of Petri nets is presented as introductory example and motivation.

Contents

1. Introduction and Related Work

1.1. Historical Background of Amalgamation

The concepts of adhesive (Lack and Sobociński, 2005) and weak adhesive high-level replacement (HLR) (Ehrig *et al.*, 2006) categories have been a break-through for the double-pushout approach of algebraic graph transformations (Rozenberg, 1997). Almost all main results for graph transformation systems could be formulated and proven in these categorical frameworks and instantiated to a large variety of HLR systems, including different kinds of graph and Petri net transformation systems (Ehrig *et al.*, 2006). These main results include the Local Church–Rosser, Parallelism, and Concurrency Theorems, the Embedding and Extension Theorem, completeness of critical pairs, and the Local Confluence Theorem (Ehrig *et al.*, 2010b, 2012). In (Ehrig *et al.*, 2010a) it is shown that also $\mathcal{M}$-adhesive categories, a slightly weaker version, are sufficient to formulate graph transformations in such a general categorical setting.

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

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

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

1.2. Other Parallel Models of Computation in Graph Transformation

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

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

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

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

1.3. Applications of Amalgamation

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

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

1.4. The Aim of this Paper

In most of the applications we need amalgamation for n rules, called multi-amalgamation, based not only on standard graph rules, but on different kinds of typed and attributed graph rules including (nested) application conditions. While some of the tools provide an ad-hoc implementation of multi-amalgamation, the underlying theory is not elaborated. The main idea of this paper is to fill this gap between theory and applications. For this purpose, we have developed the theory of multi-amalgamation for $\mathcal{M}$-adhesive systems based on rules with nested application conditions. In Ehrig *et al.* (2012), the amalgamation of exactly two rules is shortly described in this framework. Our work in this paper allows to instantiate the theory to a large variety of graphs and corresponding graph transformation systems and, using weak adhesive HLR categories, also to typed attributed graph transformation systems (Ehrig *et al.*, 2006).

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

1.5. Organization of this Paper

This paper is organized as follows. In Section 2, we discuss how to define the semantics of Petri nets using graph transformation and show, that amalgamation eases the rule definition without the need for additional control structure. In Section 3, we review basic notions of $\mathcal{M}$-adhesive categories, transformations, and application conditions. In Section 4, we introduce kernel rules, multi rules, and kernel morphisms leading to the Complement Rule Theorem as first main result. In Section 5, we construct multi-amalgamated rules and transformations and show as second main result the Multi-Amalgamation Theorem. Maximal matchings, which are used to compute the maximal amalgamated rule, are constructed in Section 6. In Section 7, we present a summary of our results and discuss future work. All more complex proofs can be found in Appendix A, while in Appendix B some technical lemmas underlying these proofs are shown.

2. Firing Semantics of Petri Nets Using Amalgamation

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

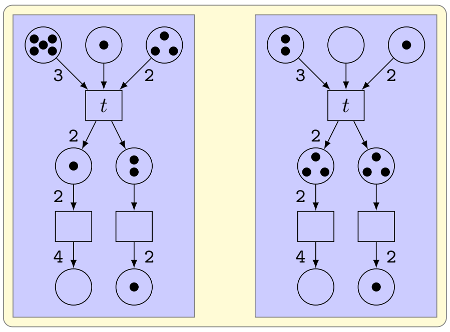


Fig. 1. The firing of the transition t

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

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

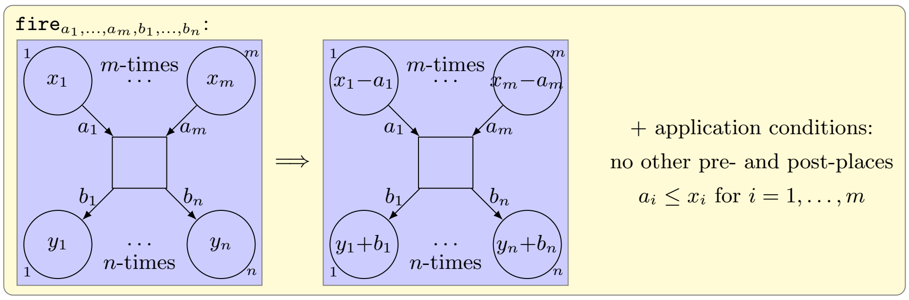


Fig. 2. The rule scheme for firing an arbitrary transition in place/transition nets

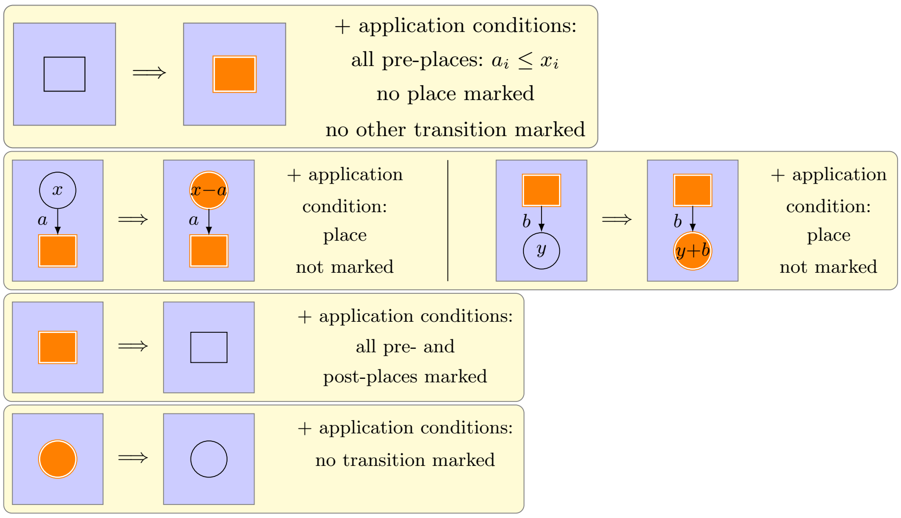


Fig. 3. The general rules for firing a transition in place/transition nets

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

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

Even for Petri nets, whose semantics can be described relatively easy on a set-theoretical basis (Reisig and Rozenberg, 1998), both graph transformation approaches discussed above have their drawbacks. When we analyze the second approach, it is obvious that the marking of the transition represents a kind of synchronization: instead of arbitrary matches for the transition, the handling of the pre and post places has to happen at this marked transition. The marking of the pre and post places is necessary to avoid multiple processing of the same place. Both markings are not necessary for the first approach, because there all places are handled at the same time. Our goal is to combine

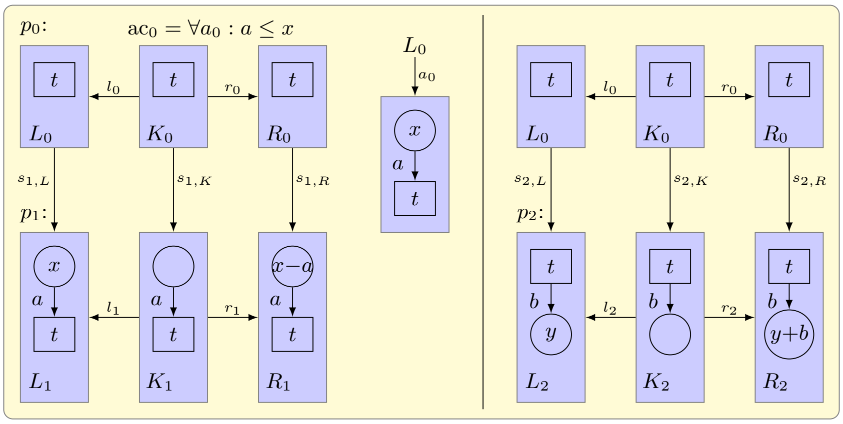


Fig. 4. The amalgamation semantics for firing place/transition nets

both approaches leading to a universal rule application for all model instances with less additional structure such that analysis becomes easier. With amalgamation, we define an interaction scheme which provides the necessary rules. Using maximal matchings the semantical step for each model instance can be computed. As shown in the following, for place/transition nets we only need one kernel rule and two multi rules to describe the complete firing semantics for all well-defined nets. We neither need infinitely many rules, which are difficult to analyze, nor any control or helper structure when using amalgamation. This eases the modeling of the semantics and prevents errors.

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

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

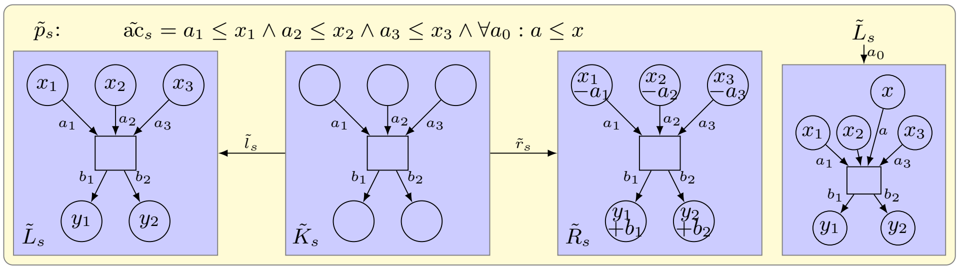


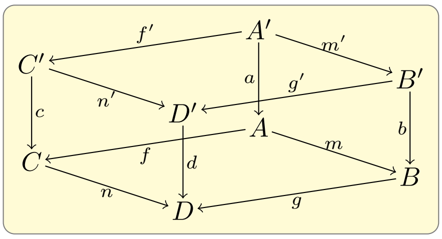
Fig. 5. The amalgamated rule for firing the transition t

describing the complete firing of t. Note that this rule looks similar to an instantiation of the rule scheme in Fig. 2, but is obtained by a very different construction mechanism, namely 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 (see (Lack and Sobociński, 2005; Ehrig *et al.*, 2006)), including the standard category of graphs which is used as a running example in this paper. $\mathcal{M}$-adhesive categories include a distinguished morphism class $\mathcal{M}$ of monomorphisms and extend adhesive categories with suitable properties. As a main difference, 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 on the right with $m \in \mathcal{M}$ is a *vertical weak van Kampen square*, short $\mathcal{M}$-van Kampen square, if it satisfies the *vertical weak van Kampen property*, i.e., for any commutative cube, where the back faces are pullbacks and the vertical morphisms $b, c, d \in \mathcal{M}$, the following statement holds: The top face is a pushout if and only if the front faces are pullbacks.



In contrast, the horizontal weak van Kampen property assumes that $f \in \mathcal{M}$ 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* $(\mathbf{C}, \mathcal{M})$ consists of a category $\mathbf{C}$ and a class $\mathcal{M}$ of monomorphisms in $\mathbf{C}$, which 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 $\mathbf{C}$ 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 $(\mathbf{Sets}, \mathcal{M})$ of sets, $(\mathbf{Graphs}, \mathcal{M})$ of graphs, $(\mathbf{Graphs_{TG}}, \mathcal{M})$ of typed graphs, $(\mathbf{ElemNets}, \mathcal{M})$ of elementary Petri nets, $(\mathbf{PTNets}, \mathcal{M})$ of place/transition nets, where for all these categories $\mathcal{M}$ is the class of all monomorphisms, and $(\mathbf{AGraphs_{ATG}}, \mathcal{M})$ of typed attributed graphs, where $\mathcal{M}$ is the class of all injective typed attributed graph morphisms with isomorphic data type component (see (Ehrig *et al.*, 2006)).

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

Definition 3.3 (Rule and transformation). A *rule* is given by a span $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ with objects L, K, and R, called left-hand side, interface, and right-hand side, respectively, and $\mathcal{M}$-morphisms l and r. An application of such a rule to an object G via a match $m : L \to G$ is constructed as two pushouts (1) and (2) leading to a *direct transformation* $G \xRightarrow{p,m} H$.

$$\begin{tikzcd} p: L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "k"] \arrow[dr, phantom, "(2)"] & R \arrow[d, "n"] \\ G & D \arrow[l, "f"] \arrow[r, "g"'] & H \end{tikzcd}$$

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

An important extension is the use of rules with suitable application conditions. These include positive application conditions of the form $\exists a$ for a morphism $a : L \to C$, demanding a certain structure in addition to L, and also negative application conditions $\neg\exists a$, forbidding such a structure. A match $m : L \to G$ satisfies $\exists a$ (resp. $\neg\exists a$) if there is a (resp. no) $\mathcal{M}$-morphism $q : C \to G$ satisfying $q \circ a = m$. In more detail, we use nested application conditions (Habel and Pennemann, 2009), short application conditions.

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

$$\begin{tikzcd} \mathrm{ac} \vartriangleright L \arrow[rr, "a"] \arrow[dr, "m"'] & & C \vartriangleleft \mathrm{ac}' \arrow[dl, "q"] \\ & G & \end{tikzcd}$$

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

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

In this paper we consider rules of the form $p = (L \xleftarrow{l} K \xrightarrow{r} R, \mathrm{ac})$, where $(L \xleftarrow{l} K \xrightarrow{r} R)$ is a (plain) rule and ac is an application condition on L. In order to handle rules with application conditions there are two important concepts, called the shift of application conditions over morphisms and rules (Habel and Pennemann, 2009; Ehrig *et al.*, 2010b).

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

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

$$
\begin{tikzcd}
\mathrm{ac} \rhd & P \arrow[r, "a"] \arrow[d, "b"'] & C \arrow[d, "b'"] & \lhd\ \mathrm{ac}' \\
\mathrm{Shift}(b, \mathrm{ac}) \rhd & P' \arrow[r, "a'"'] & C' & \lhd\ \mathrm{Shift}(b', \mathrm{ac}')
\end{tikzcd}
$$

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

Example 3.3. An example for shifting an application condition over a morphism can be found in the left of Fig. 7. We have that $\mathrm{Shift}(v_1, \neg\exists a_1') = \neg\exists a_{11}$, because square $(*)$ is the only possible commuting square leading to a_{11}, b_{11} jointly surjective and b_{11} injective.

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

$$
\begin{tikzcd}
\mathrm{ac} \rhd & P \arrow[rr, "b"] \arrow[dr, "p \circ b"'] & & P' \arrow[dl, "p"] & \lhd\ \mathrm{Shift}(b, \mathrm{ac}) \\
& & G & &
\end{tikzcd}
$$

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

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

Definition 3.6 (Shift over rule). Given a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R, \mathrm{ac})$ and an application condition $\mathrm{ac}_R = \exists(a, \mathrm{ac}_R')$ over R, then $\mathrm{L}(p, \mathrm{ac}_R)$ is an application condition over L defined by $\mathrm{L}(p, \mathrm{ac}_R) = \exists(b, \mathrm{L}(p^*, \mathrm{ac}_R'))$ if $a \circ r$ has a pushout complement (1) and $p^* = (Y \overset{l^*}{\leftarrow} Z \overset{r^*}{\rightarrow} X)$ is the derived rule by constructing pushout (2), otherwise false. Moreover, $\mathrm{L}(p, \mathrm{true}) = \mathrm{true}$ and the construction is extended to Boolean formulas in the usual way.

$$
\begin{tikzcd}
\mathrm{L}(p, \mathrm{ac}_R) \rhd & L \arrow[d, "b"'] \arrow[dr, phantom, "(2)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "c"'] \arrow[dr, phantom, "(1)"] & R \arrow[d, "a"] & \lhd\ \mathrm{ac}_R \\
\mathrm{L}(p^*, \mathrm{ac}_R') \rhd & Y & Z \arrow[l, "l^*"] \arrow[r, "r^*"'] & X & \lhd\ \mathrm{ac}_R'
\end{tikzcd}
$$

Example 3.4. An example for shifting an application condition over a rule shown by the two pushout squares (PO_1) and (PO_2) in Fig. 7, where $\mathrm{L}(p_1^*, \neg\exists a_{11}) = \neg\exists a_1$.

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

$$\begin{tikzcd}
\mathrm{L}(p,\mathrm{ac}_R) \triangleright & L \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "k"] \arrow[dr, phantom, "(2)"] & R \arrow[d, "n"] & \triangleleft\ \mathrm{ac}_R \\
& G & D \arrow[l, "f"] \arrow[r, "g"'] & H &
\end{tikzcd}$$

Proof. See (Habel and Pennemann, 2009). □

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

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

— $\mathrm{Shift}(b, \mathrm{Shift}(a, \mathrm{ac})) \cong \mathrm{Shift}(b \circ a, \mathrm{ac})$,

— $\mathrm{Shift}(m, \mathrm{L}(p, \mathrm{ac})) \cong \mathrm{L}(p', \mathrm{Shift}(n, \mathrm{ac}))$.

$$\begin{tikzcd}
& & & \mathrm{ac}\ \triangledown & & \\
p: & L \arrow[d, "m"'] \arrow[dr, phantom, "(2)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "k"] \arrow[dr, phantom, "(1)"] & R \arrow[d, "n"] \arrow[r, "a"] & P \arrow[r, "b"] & Q \\
p': & L' & K' \arrow[l, "l'"] \arrow[r, "r'"'] & R' & &
\end{tikzcd}$$

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

General Assumptions

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

4. Decomposition of Direct Transformations

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

A kernel morphism describes how a smaller rule, the kernel rule, is embedded into a larger rule, the multi rule, which has its name because it can be applied multiple times for a given kernel rule match as described in Section 5. We need some more technical preconditions to make sure that the embeddings of the L-, K-, and R-components as well as the application conditions are consistent and allow to construct a complement rule.

Definition 4.1 (Kernel morphism). Given rules $p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, \mathrm{ac}_0)$ and $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \mathrm{ac}_1)$, a *kernel morphism* $s_1 : p_0 \to p_1$, $s_1 = (s_{1,L}, s_{1,K}, s_{1,R})$ consists of $\mathcal{M}$-morphisms $s_{1,L} : L_0 \to L_1$, $s_{1,K} : K_0 \to K_1$, and $s_{1,R} : R_0 \to R_1$ such that in the following diagram (1_1) and (2_1) are pullbacks, (1_1) has a pushout complement $(1'_1)$ for $s_{1,L} \circ l_0$, and ac_0 and ac_1 are *complement-compatible w.r.t.* s_1, i.e. given pushout (3_1) then $\mathrm{ac}_1 \cong \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(v_1, \mathrm{ac}'_1))$ for some ac'_1 on L_{10} and $p_1^* = (L_1 \xleftarrow{u_1} L_{10} \xrightarrow{v_1} E_1)$. In this case, p_0 is called *kernel rule* and p_1 *multi rule*.

$$
\begin{tikzcd}
p_0 : \mathrm{ac}_0 \triangleright \arrow[d, "s_1"'] & L_0 \arrow[d, "s_{1,L}"'] \arrow[dr, phantom, "(1_1)"] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "s_{1,K}"] \arrow[dr, phantom, "(2_1)"] & R_0 \arrow[d, "s_{1,R}"] \\
p_1 : \mathrm{ac}_1 \triangleright & L_1 & K_1 \arrow[l, "l_1"] \arrow[r, "r_1"'] & R_1
\end{tikzcd}
\qquad
\begin{tikzcd}
L_0 \arrow[d, "s_{1,L}"'] \arrow[dr, phantom, "(1'_1)"] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "w_1"] \arrow[dr, phantom, "(3_1)"] & R_0 \arrow[d, "e_{11}"] \\
L_1 & L_{10} \arrow[l, "u_1"] \arrow[r, "v_1"'] & E_1 \\
 & \mathrm{ac}'_1 \triangle &
\end{tikzcd}
$$

Remark 4.1. The complement-compatibility of the application conditions makes sure that there is a decomposition of ac_1 into parts on L_0 and L_{10}, where the latter ones are used later for the application conditions of the complement rule.

Example 4.1. To explain the concept of amalgamation, in our example we model a small transformation system for switching the direction of edges in labeled graphs, where we only have different labels for edges – black and dotted edges. The kernel rule p_0 is

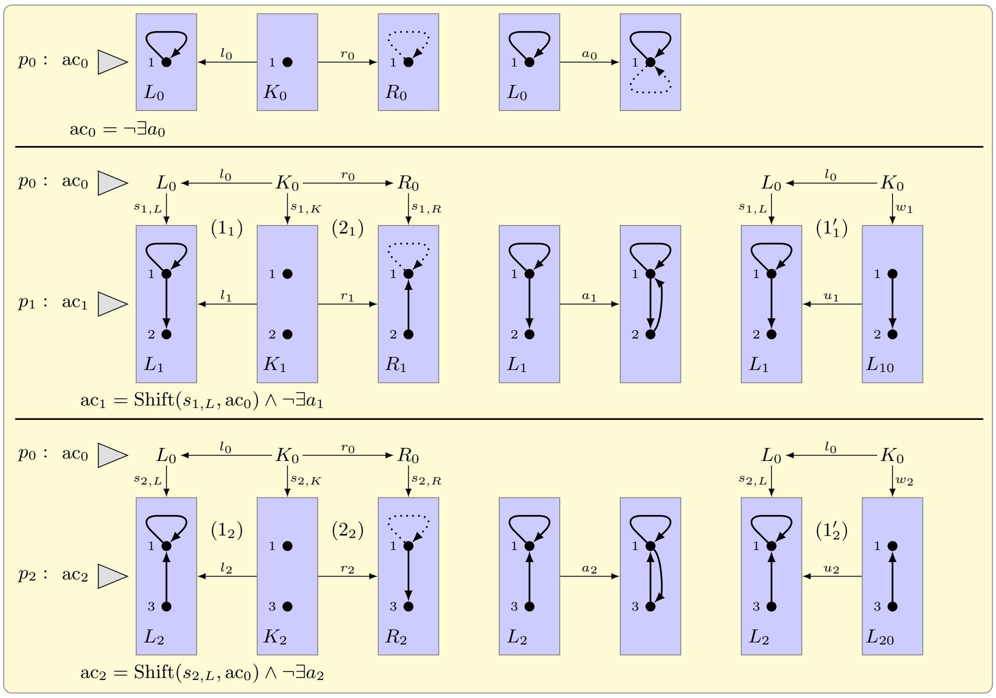


Fig. 6. The kernel rule p_0 and the multi rules p_1 and p_2

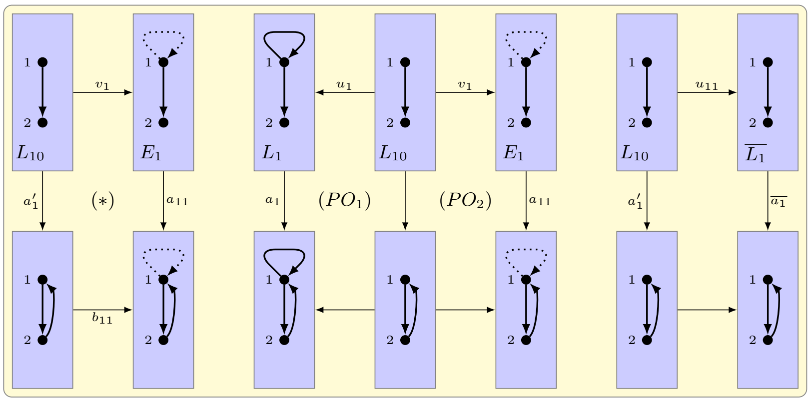


Fig. 7. Constructions for the application conditions

depicted in the top of Fig. 6. It selects a node with a black loop, deletes this loop, and adds a dotted loop, all of this if no dotted loop is already present. The matches are defined by the numbers at the nodes and can be induced for the edges by their position.

In the middle and bottom of Fig. 6, two multi rules p_1 and p_2 are shown, which extend the rule p_0 and in addition reverse an edge if no backward edge is present. They also inherit the application condition of p_0 forbidding a dotted loop at the selected node. There is a kernel morphism $s_1 : p_0 \to p_1$ as shown in the top of Fig. 6 with pullbacks (1_1) and (2_1), and pushout complement $(1_1')$. For the application conditions, $\mathrm{ac}_1 = \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \neg\exists a_1 \cong \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(v_1, \neg\exists a_1'))$ as shown in the left of Fig. 7. Thus $\mathrm{ac}_1' = \neg\exists a_1'$, and ac_0 and ac_1 are complement compatible.

Similarly, there is a kernel morphism $s_2 : p_0 \to p_2$ as shown in the bottom of Fig. 6 with pullbacks (1_2) and (2_2), pushout complement $(1_2')$, and ac_0 and ac_2 are complement compatible.

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

Theorem 4.1 (Existence of complement rule). Given rules $p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, \mathrm{ac}_0)$ and $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \mathrm{ac}_1)$, and a kernel morphism $s_1 : p_0 \to p_1$ then there is a canonical way to construct a rule $\overline{p_1} = (\overline{L_1} \xleftarrow{\overline{l_1}} \overline{K_1} \xrightarrow{\overline{r_1}} \overline{R_1}, \overline{\mathrm{ac}_1})$ and a jointly epimorphic cospan $R_0 \xrightarrow{e_{11}} E_1 \xleftarrow{e_{12}} \overline{L_1}$ such that the E_1-concurrent rule $p_0 *_{E_1} \overline{p_1}$ exists

$$\begin{tikzcd}
 & & & & & \overline{\mathrm{ac}_1}\,\triangledown & & \\
\mathrm{ac}_0 \arrow[r, phantom, "\triangleright"] & L_0 \arrow[d, "s_{1,L}"'] \arrow[r, phantom, "(1_1')", shift right=4] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "w_1"'] \arrow[r, phantom, "(3_1)", shift right=6] & R_0 \arrow[dr, "e_{11}"'] & & \overline{L_1} \arrow[dl, "e_{12}"] \arrow[r, phantom, "(9_1)", shift right=6] & \overline{K_1} \arrow[l, "\overline{l_1}"'] \arrow[r, "\overline{r_1}"] \arrow[d, "\overline{w_1}"] \arrow[r, phantom, "(13_1)", shift right=6] & \overline{R_1} \arrow[d, "t_1"] \\
\mathrm{ac}_1 \arrow[r, phantom, "\triangleright"] & L_1 & L_{10} \arrow[l, "u_1"'] \arrow[rr, "v_1"] & & E_1 \arrow[d, phantom, "(8_1)+(9_1)"] & & R_{10} \arrow[ll, "\overline{u_1}"'] \arrow[r, "\overline{v_1}"] & R_1 \\
 & & \triangle\,\mathrm{ac}_1' & & K_1 \arrow[ulll, "l_1"] \arrow[ull, "l_{10}"'] \arrow[urr, "r_{10}"] \arrow[urrr, "r_1"'] & & &
\end{tikzcd}$$

and $p_1 = p_0 *_{E_1} \overline{p_1}$. For the definition of E-concurrent rules for rules with application conditions see (Ehrig *et al.*, 2010b).

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

Remark 4.2. Note, that using the construction in the appendix the interface K_0 of the kernel rule has to be preserved in the complement rule. This canonical construction of $\overline{p_1}$ is not unique w.r.t. the property $p_1 = p_0 *_{E_1} \overline{p_1}$, since other choices for S_1 with $\mathcal{M}$-morphisms s_{11} and s_{13} also lead to a well-defined construction. In particular, one could choose $S_1 = R_0$ leading to $\overline{p_1} = E_1 \xleftarrow{\overline{u_1}} R_{10} \xrightarrow{\overline{v_1}} R_1$. Our choice represents a smallest possible complement, which should be preferred in most application areas.

Definition 4.2 (Complement rule). Given rules $p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, \mathrm{ac}_0)$ and $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \mathrm{ac}_1)$, and a kernel morphism $s_1 : p_0 \to p_1$ then the canonical rule $\overline{p_1} = (\overline{L_1} \xleftarrow{\overline{l_1}} \overline{K_1} \xrightarrow{\overline{r_1}} \overline{R_1}, \overline{\mathrm{ac}_1})$ identified by Thm. 4.1 is called *complement rule* (of s_1).

Example 4.2. Consider the kernel morphism s_1 depicted in Fig. 6. Using the construction in Thm. 4.1 we obtain the complement rule in the top row in Fig. 8 with the application condition $\overline{\mathrm{ac}_1} = \neg\exists\overline{a_1}$ constructed in the right of Fig. 7. In Fig. 9, the diagrams of the construction are shown. Similarly, we obtain a complement rule for the kernel morphism $s_2 : p_0 \to p_2$ in Fig. 6, which is depicted in the bottom row of Fig. 8.

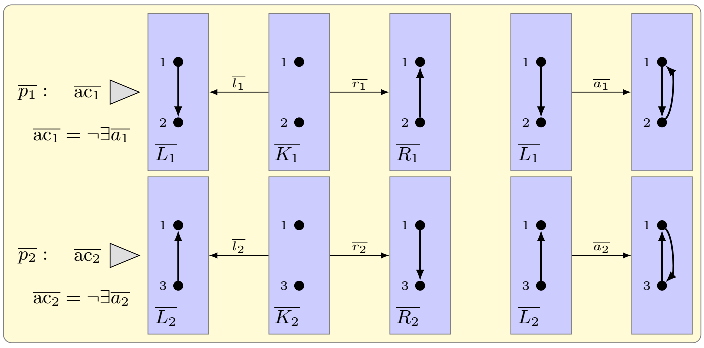


Fig. 8. The complement rules for the kernel morphisms

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

Fact 4.1 (Decomposition of transformations). Given rules $p_0 = (L_0 \xleftarrow{l_0} K_0 \xrightarrow{r_0} R_0, \mathrm{ac}_0)$ and $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1, \mathrm{ac}_1)$, a kernel morphism $s_1 : p_0 \to p_1$, and a direct transformation $t_1 : G \xRightarrow{p_1, m_1} G_1$ then t_1 can be decomposed into the transformation $G \xRightarrow{p_0, m_0} G_0 \xRightarrow{\overline{p_1}, \overline{m_1}} G_1$ with $m_0 = m_1 \circ s_{1,L}$ where $\overline{p_1}$ is the complement rule of s_1.

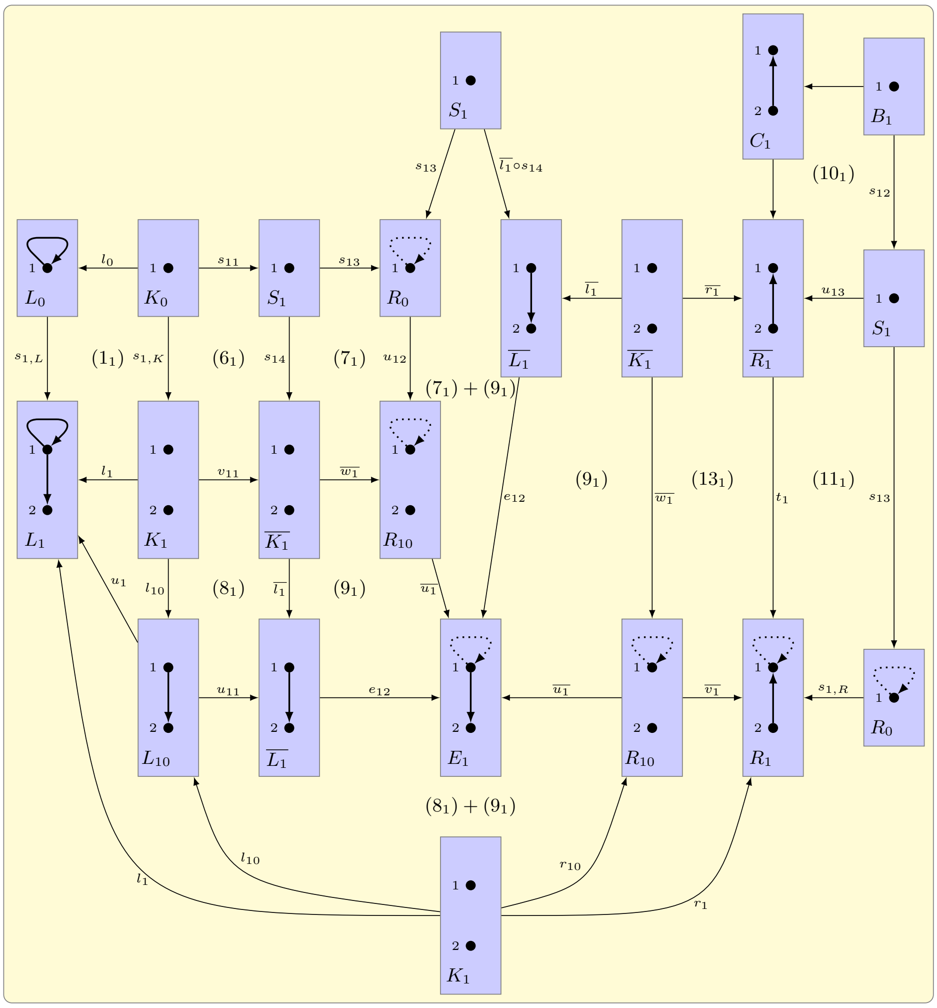


Fig. 9. The construction of the complement rule for the kernel morphism s_1

$$\begin{tikzcd} G \arrow[rr, Rightarrow, "p_1, m_1"] \arrow[dr, Rightarrow, "p_0, m_0"'] & & G_1 \\ & G_0 \arrow[ur, Rightarrow, "\overline{p_1}, \overline{m_1}"'] & \end{tikzcd}$$

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

5. Multi-Amalgamation

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

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

Definition 5.1 (Amalgamated rule). Given rules $p_i = (L_i \xleftarrow{l_i} K_i \xrightarrow{r_i} R_i, \mathrm{ac}_i)$ for $i = 0, \dots, n$ and a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,\dots,n}$, then the *amalgamated rule* $\tilde{p}_s = (\tilde{L}_s \xleftarrow{\tilde{l}_s} \tilde{K}_s \xrightarrow{\tilde{r}_s} \tilde{R}_s, \tilde{\mathrm{ac}}_s)$ is constructed as the componentwise colimit of the kernel morphisms:

$$\begin{tikzcd}
p_0 : \mathrm{ac}_0 \arrow[d, "s_i"'] & \rhd & L_0 \arrow[d, "s_{i,L}"'] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "s_{i,K}"'] \arrow[l, phantom, "(1_i)", shift right=6] & R_0 \arrow[d, "s_{i,R}"'] \arrow[l, phantom, "(2_i)", shift right=6] \\
p_i : \mathrm{ac}_i \arrow[d, "t_i"'] & \rhd & L_i \arrow[d, "t_{i,L}"'] & K_i \arrow[l, "l_i"] \arrow[r, "r_i"'] \arrow[d, "t_{i,K}"'] \arrow[l, phantom, "(14_i)", shift right=6] & R_i \arrow[d, "t_{i,R}"'] \arrow[l, phantom, "(15_i)", shift right=6] \\
\tilde{p}_s : \tilde{\mathrm{ac}}_s & \rhd & \tilde{L}_s & \tilde{K}_s \arrow[l, "\tilde{l}_s"] \arrow[r, "\tilde{r}_s"'] & \tilde{R}_s
\end{tikzcd}$$

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

Fact 5.1. The amalgamated rule is well-defined and we have kernel morphisms $t_i = (t_{i,L}, t_{i,K}, t_{i,R}) : p_i \to \tilde{p}_s$ for $i = 0, \dots, n$.

Proof. See Subsection A.2 in the appendix. □

The application of an amalgamated rule yields an amalgamated transformation.

Definition 5.2 (Amalgamated transformation). The application of an amalgamated rule to a graph G is called an *amalgamated transformation.*

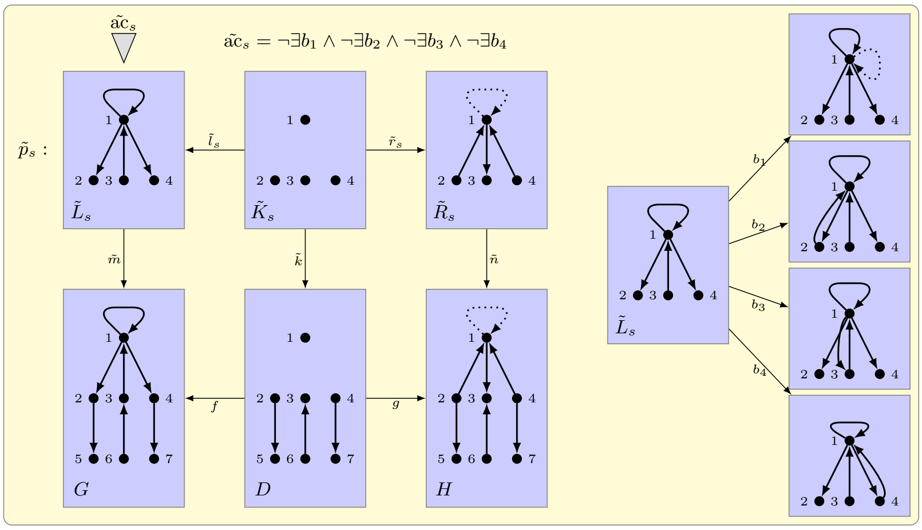


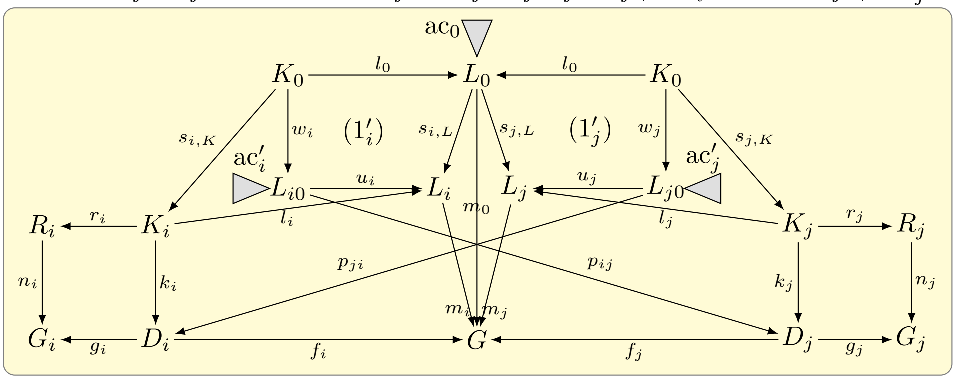
Fig. 10. An amalgamated transformation

Example 5.1. Consider the bundle $s = (s_1, s_2, s_3 = s_1)$ of the kernel morphisms depicted in Fig. 6. The corresponding amalgamated rule $\tilde{p}_s$ is shown in the top row of Fig. 10. This amalgamated rule can be applied to the graph G leading to the amalgamated transformation depicted in Fig. 10, where the application condition $\tilde{ac}_s$ is obviously fulfilled by the match $\tilde{m}$.

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

Definition 5.3 (s-amalgamable). Given a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,\ldots,n}$, a bundle of direct transformations steps $(G \xRightarrow{p_i, m_i} G_i)_{i=1,\ldots n}$ is s-*amalgamable*, if

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



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

Fact 5.2. For graphs and other set-based structures, weakly independent matches without considering the application conditions means that $m_i(L_i) \cap m_j(L_j) \subseteq m_0(L_0) \cup (m_i(l_i(K_i)) \cap m_j(l_j(K_j)))$ for all $i \neq j$, i.e. the elements in the intersection of the matches m_i and m_j are either preserved by both transformations, or are also matched by m_0.

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

$$\begin{tikzcd}
 & & L_0 \arrow[dl, "s_{i,L}"'] \arrow[dr, "s_{j,L}"] \arrow[dd, "m_0"] & & \\
K_i \arrow[r, "l_i"] & L_i \arrow[dr, "m_i"'] & & L_j \arrow[dl, "m_j"] & K_j \arrow[l, "l_j"'] \\
 & & G & &
\end{tikzcd}$$

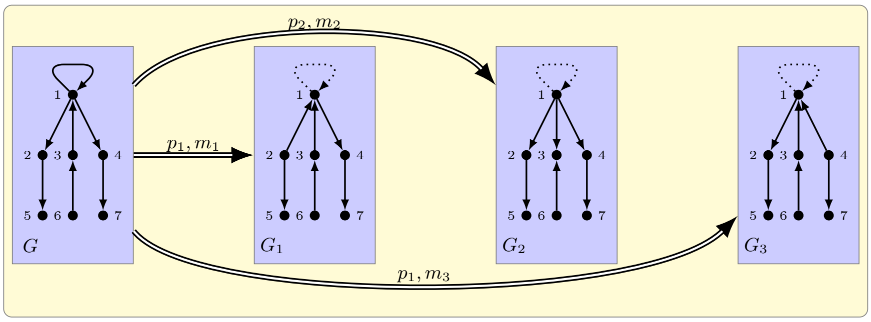


Fig. 11. An s-amalgamable bundle of direct transformations

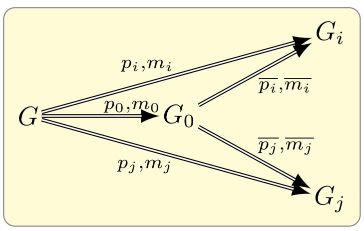
"$\Leftarrow$" Let $x = m_i(y_i) = m_j(y_j)$, and suppose $x \notin m_0(L_0)$. Since $(1'_i)$ is a pushout we have that $y_i = u_i(z_i) \in u_i(L_{i0} \backslash w_i(K_0))$, and $x = m_i(u_i(z_i)) = f_j(p_{ij}(z_i)) = m_j(y_j)$, and by pushout properties $y_j \in l_j(K_j)$ and $x \in m_j(l_j(K_j))$.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

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

Example 5.2. Consider the bundle $s = (s_1, s_2, s_3 = s_1)$ of kernel morphisms from Ex. 5.1. For the graph G given in Fig. 10 we find matches $m_0 : L_0 \to G$, $m_1 : L_1 \to G$, $m_2 : L_2 \to G$, and $m_3 : L_1 \to G$ mapping all nodes from the left hand side to their corresponding nodes in G, except for m_3 mapping node 2 in L_1 to node 4 in G. For all these matches, the corresponding application conditions are fulfilled and we can apply the rules p_1, p_2, p_1, respectively, leading to the bundle of direct transformations depicted in Fig. 11. This bundle is s-amalgamable, because the matches m_1, m_2, and m_3 agree on the match m_0, and are weakly independent, because they only overlap in m_0.

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

Fact 5.3. Given a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,\ldots,n}$ and an s-amalgamable bundle of direct transformations $(G \xRightarrow{p_i, m_i} G_i)_{i=1,\ldots,n}$ then each direct transformation $G \xRightarrow{p_i, m_i} G_i$ can be decomposed into a transformation $G \xRightarrow{p_0, m_0} G_0 \xRightarrow{\overline{p_i}, \overline{m_i}} G_i$. Moreover, the transformations $G_0 \xRightarrow{\overline{p_i}, \overline{m_i}} G_i$ are pairwise parallel independent.



Proof. See Subsection A.3 in the appendix. □

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

Theorem 5.1 (Multi-Amalgamation). Consider a bundle of kernel morphisms $s = (s_i : p_0 \to p_i)_{i=1,\dots,n}$.

$$\begin{tikzcd} & G_i \arrow[dr, Rightarrow, "q_i"] & \\ G \arrow[ur, Rightarrow, "p_i, m_i"] \arrow[rr, Rightarrow, "\tilde{p}_s, \tilde{m}"'] & & H \end{tikzcd}$$

1 *Synthesis.* Given an s-amalgamable bundle $(G \xRightarrow{p_i,m_i} G_i)_{i=1,\dots,n}$ of direct transformations then 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 $G \xRightarrow{p_i,m_i} G_i \xRightarrow{q_i} H$ is a decomposition of $G \xRightarrow{\tilde{p}_s,\tilde{m}} H$.

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

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

Proof. See Subsection A.4 in the appendix. $\square$

Remark 5.1. Note, that q_i can be constructed as the amalgamated rule of the kernel morphisms $(p_{K_0} \to \overline{p_j})_{j \neq i}$, where $p_{K_0} = (K_0 \xleftarrow{id_{K_0}} K_0 \xrightarrow{id_{K_0}} K_0, \mathrm{true}))$ and $\overline{p_j}$ is the complement rule of p_j.

For $n = 2$ and rules without application conditions, the Multi-Amalgamation Theorem specializes to the Amalgamation Theorem in (Böhm *et al.*, 1987). Moreover, if p_0 is the empty rule, this is the Parallelism Theorem in (Ehrig *et al.*, 2010b), since the transformations are parallel independent for an empty kernel match.

Example 5.3. As already stated in Example 5.2, the transformations $G \xRightarrow{p_1,m_1} G_1$, $G \xRightarrow{p_2,m_2} G_2$, and $G \xRightarrow{p_1,m_3} G_3$ shown in Fig. 11 are s-amalgamable for the bundle $s = (s_1, s_2, s_3 = s_1)$ of kernel morphisms. Applying Fact 5.3, we can decompose these transformations into a transformation $G \xRightarrow{p_0,m_0} G_0$ followed by transformations

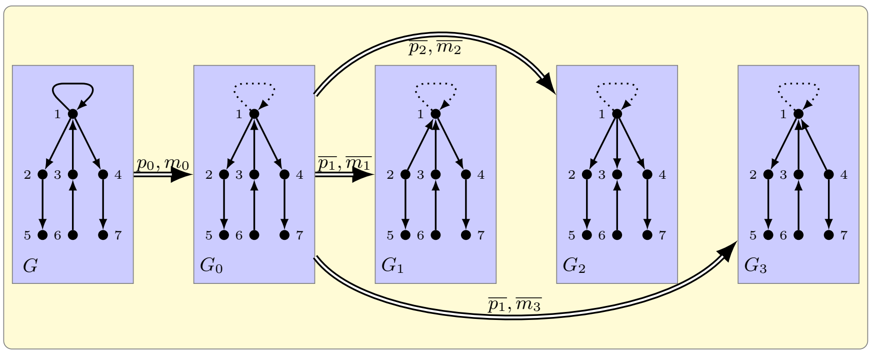


Fig. 12. The decomposition of the s-amalgamable bundle

$G_0 \xRightarrow{\overline{p_1},\overline{m_1}} G_1$, $G_0 \xRightarrow{\overline{p_2},\overline{m_2}} G_2$, and $G_0 \xRightarrow{\overline{p_1},\overline{m_3}} G_3$ via the complement rules, which are pairwise parallel independent. These transformations are depicted in Fig. 12. Moreover, Thm. 5.1 implies that we obtain for this bundle of direct transformations an amalgamated transformation $G \xRightarrow{\tilde{p}_s,\tilde{m}} H$, which is the transformation already shown in Fig. 10. Vice versa, the analysis of this amalgamated transformation leads to the s-amalgamable bundle of transformations $G \xRightarrow{p_1,m_1} G_1$, $G \xRightarrow{p_2,m_2} G_2$, and $G \xRightarrow{p_1,m_3} G_3$ from Fig. 11.

6. Multi-Amalgamation with Maximal Matchings

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

An interaction scheme defines a bundle of kernel morphisms. In contrast to a concrete bundle, for the application of such an interaction scheme all possible matches for the multi rules are computed that agree on a given kernel match 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 $s_i : p_0 \to p_i$ form an *interaction scheme* $is = \{s_1, \ldots, s_k\}$.

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

Definition 6.2 (Maximal weakly independent matching). Given an interaction scheme $is = \{s_1, \ldots, s_k\}$ 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 $G \xRightarrow{p'_i,m_i} G_i$, then 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 $((m_i), m')$ fulfills this property.

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

Algorithm 6.1 (Maximal weakly independent matching). Given a graph G and an interaction scheme $is = \{s_1, \ldots, s_k\}$, a maximal weakly disjoint matching $m = (m_0, m_1, \ldots, m_n)$ can be computed as follows:

1 Set $i = 0$. Choose a kernel matching $m_0 : L_0 \to G$ such that $G \xRightarrow{p_0, m_0} G_0$ is a valid transformation.
2 As long as possible: Increase i, choose a multi rule $\hat{p}_i = p_j$ with $j \in \{1, \ldots, k\}$, and find a match $m_i : L_j \to G$ such that $m_i \circ s_{j,L} = m_0$, $G \xRightarrow{p_j, m_i} G_i$ is a valid transformation, the matches $m_1, \ldots, m_i$ are weakly independent, and $m_i \neq m_\ell$ for all $\ell = 1, \ldots, i-1$.
3 If no more valid matches for any rule in the interaction scheme can be found, return $m = (m_0, m_1, \ldots, m_n)$.

The maximal weakly independent matching leads to a bundle of kernel morphisms $s = (s_i : p_0 \to \hat{p}_i)$ 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 needs a lot of backtracking, because often a match in Step 2 is not weakly independent from an already chosen one, which has to be checked pairwise for this new match compared to all others. While the application conditions always have to be analyzed, since they may state global properties of the resulting graph, at least for the elements available for the new match some restrictions may help to enhance the computation. In many cases, it is adequate to require the matches to be disjoint outside the kernel match. A typical example is the semantics of Petri nets as described in Section 2 - there, all maximal weakly independent matchings are also weakly disjoint. This disjointness property is described formally by a certain pullback requirement. Using maximal weakly disjoint matchings for implementation, we can rule out model parts that were already matched.

Definition 6.3 (Maximal weakly disjoint matching). Given an interaction scheme $is = \{s_1, \ldots, s_k\}$ and a maximal weakly independent matching $m = (m_i : L'_i \to G)$ then m forms a *maximal weakly disjoint matching* if the square $(P_{i\ell})$ is a pullback for all $i \neq \ell$.

$$\begin{tikzcd} L_0 \arrow[r, "s_{i,L}"] \arrow[d, "s_{\ell,L}"'] \arrow[dr, phantom, "(P_{i\ell})"] & L'_i \arrow[d, "m_i"] \\ L'_\ell \arrow[r, "m_\ell"'] & G \end{tikzcd}$$

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

Fact 6.1. Given an object G, a bundle of kernel morphisms $s = (s_1, \ldots, s_n)$, and matches $m_1, \ldots, m_n$ leading to a bundle of direct transformations $G \xRightarrow{p_i, m_i} G_i$ such that $m_i \circ s_{i,L} = m_0$ and square $(P_{i\ell})$ is a pullback for all $i \neq \ell$ then 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. It 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. For the cube on the right, the bottom face is pushout (20_i), the back right face is pullback (1_i), and the front right face is pullback $(P_{i\ell})$. Now construct the pullback of f_i and m_ℓ as the front left face, and from $m_\ell \circ s_{\ell,L} \circ l_0 = m_i \circ s_{i,L} \circ l_0 = m_i \circ l_i \circ s_{i,K} = f_i \circ k_i \circ s_{i,K}$ we obtain a morphism p with $\hat{f} \circ p = s_{\ell,L} \circ l_0$ and $\hat{m} \circ p = k_i \circ s_{i,K}$.

$$\begin{tikzcd}
L_i \arrow[d, "m_i"'] \arrow[dr, phantom, "(20_i)"] & K_i \arrow[l, "l_i"'] \arrow[r, "r_i"] \arrow[d, "k_i"] \arrow[dr, phantom, "(21_i)"] & R_i \arrow[d, "n_i"] \\
G & D_i \arrow[l, "f_i"] \arrow[r, "g_i"'] & G_i
\end{tikzcd}
\qquad
\begin{tikzcd}
P \arrow[dr, "\hat{f}"] \arrow[dd, "\hat{m}"'] & & K_0 \arrow[ll, "p"'] \arrow[rr, "l_0"] \arrow[d, "s_{i,K}"] & & L_0 \arrow[dlll, "s_{\ell,L}"'] \arrow[dd, "s_{i,L}"] \\
& L_\ell \arrow[ddr, "m_\ell"] & K_i \arrow[dll, "k_i"'] \arrow[drr, "l_i"] & & \\
D_i \arrow[drr, "f_i"'] & & & & L_i \arrow[dll, "m_i"] \\
& & G & &
\end{tikzcd}$$

From pullback composition and decomposition of the right and left faces it follows that also the back left face is a pullback. Now the $\mathcal{M}$-van Kampen property can be applied leading to a pushout in the top face. Since pushout complements are unique up to isomorphism, we can substitute the top face by pushout $(1'_i)$ from Def. 5.3 with $P \cong L_{\ell 0}$. Thus we have found the morphism $p_{\ell i} := \hat{m}$ with $f_i \circ p_{\ell i} = m_\ell \circ u_i$. This construction can be applied for all pairs i, ℓ leading to weakly independent matches without application conditions. $\square$

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

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

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

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

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

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

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. It leads to the bundle $s = (s_1, s_1, s_1, s_2, s_2)$ and the amalgamated rule $\tilde{p}_s$, which can be applied to X leading to the amalgamated transformation $X \xRightarrow{\tilde{p}_s, \tilde{m}} X'$ as shown in the left of Fig. 13.

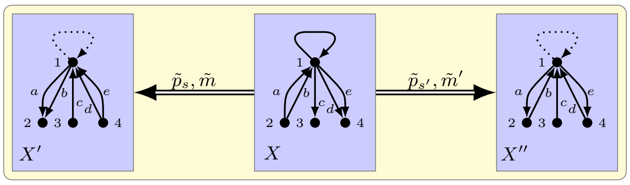


Fig. 13. Application of an amalgamated rule via maximal matchings

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

7. Conclusion

In this paper, we have generalized the theory of amalgamation in (Böhm *et al.*, 1987) to multi-amalgamation in $\mathcal{M}$-adhesive categories and introduced interaction schemes and maximal matchings. More precisely, the Complement Rule and Amalgamation Theorems in (Böhm *et al.*, 1987) are presented on a set-theoretical basis for pairs of plain graph rules without any application conditions. The Complement Rule and Multi-Amalgamation Theorems in this paper are valid in adhesive and $\mathcal{M}$-adhesive categories for n rules with application conditions (Habel and Pennemann, 2009). These generalizations are non-trivial and important for applications of parallel graph transformations to communication-based systems (Taentzer, 1996), to model transformations from BPMN to BPEL (Biermann *et al.*, 2010a), and for the modeling of the operational semantics of visual languages (Ermel, 2006), where interaction schemes are used to generate multi-amalgamated rules and transformations based on suitable maximal matchings.

The theory of multi-amalgamation is a solid mathematical basis to analyze interesting properties of the operational semantics, like termination, local confluence, and functional behavior. However, it is left open for future work to generalize the corresponding results in (Ehrig *et al.*, 2006), like the Local Church–Rosser, Parallelism, and Local Confluence Theorems, to the case of multi-amalgamated rules, especially to the operational semantics of statecharts based on amalgamated graph transformation with maximal matchings in (Golas *et al.*, 2011).

References

Balasubramanian, D., Narayanan, A., Neema, S., Shi, F., Thibodeaux, R., Karsai, G., 2007. A Subgraph Operator for Graph Transformation Languages. Electronic Communications of the EASST 6, 1–12.

Biermann, E., Ehrig, H., Ermel, C., Golas, U., Taentzer, G., 2010a. Parallel Independence of Amalgamated Graph Transformations Applied to Model Transformation. In: Graph Transformations and Model-Driven Engineering. Vol. 5765 of Lecture Notes in Computer Science. Springer, pp. 121–140.

Biermann, E., Ermel, C., Taentzer, G., 2010b. Lifting Parallel Graph Transformation Concepts to Model Transformation Based on the Eclipse Modeling Framework. Electronic Communications of the EASST 26, 1–19.

Böhm, P., Fonio, H.-R., Habel, A., 1987. Amalgamation of Graph Transformations: A Synchronization Mechanism. Journal of Computer and System Sciences 34 (2-3), 377–408.

Castellani, I., Montanari, U., 1983. Graph Grammars for Distributed Systems. In: Ehrig, H., Nagl, M., Rozenberg, G. (Eds.), Graph Grammars and Their Application to Computer Science. Vol. 153 of Lecture Notes in Computer Science. Springer, pp. 20–38.

de Lara, J., Ermel, C., Taentzer, G., Ehrig, K., 2004. Parallel Graph Transformation for Model Simulation Applied to Timed Transition Petri Nets. Electronic Notes in Theoretical Computer Science 109, 17–29.

Degano, P., Montanari, U., 1987. A Model of Distributed Systems Based on Graph Rewriting. Journal of the ACM 34 (2), 411–449.

Ehrig, H., Ehrig, K., Prange, U., Taentzer, G., 2006. Fundamentals of Algebraic Graph Transformation. EATCS Monographs. Springer.

Ehrig, H., Golas, U., Habel, A., Lambers, L., Orejas, F., 2012. M-Adhesive Transformation Systems with Nested Application Conditions. Part 1: Parallelism, Concurrency and Amalgamation. Mathematical Structures in Computer Science (this volume).

Ehrig, H., Golas, U., Hermann, F., 2010a. Categorical Frameworks for Graph Transformation and HLR Systems based on the DPO Approach. Bulletin of the EATCS 102, 111–121.

Ehrig, H., Habel, A., Lambers, L., 2010b. Parallelism and Concurrency Theorems for Rules with Nested Application Conditions. Electronic Communications of the EASST 26, 1–23.

Ehrig, H., Kreowski, H.-J., 1976. Parallelism of Manipulations in Multidimensional Information Structures. In: Proceedings of MFCS 1976. Vol. 45 of Lecture Notes in Computer Science. Springer, pp. 285–293.

Ermel, C., 2006. Simulation and Animation of Visual Languages based on Typed Algebraic Graph Transformation. Ph.D. thesis, Technische Universität Berlin.

Fischer, T., Niere, J., Torunski, L., Zündorf, A., 2000. A New Graph Rewrite Language Based on the Unified Modeling Language. In: Proceedings of TAGT 1998. Vol. 1764 of Lecture Notes in Computer Science. Springer, pp. 296–309.

Golas, U., 2011. Analysis and Correctness of Algebraic Graph and Model Transformations. Ph.D. thesis, Technische Universität Berlin, Vieweg + Teubner.

Golas, U., Biermann, E., Ehrig, H., Ermel, C., 2011. A Visual Interpreter Semantics for Statecharts Based on Amalgamated Graph Transformation. Electronic Communications of the EASST 39, 1–24.

Golas, U., Ehrig, H., Habel, A., 2010. Multi-Amalgamation in Adhesive Categories. In: Graph Transformations. Proceedings of ICGT 2010. Vol. 6372 of Lecture Notes in Computer Science. Springer, pp. 346–361.

Grønmo, R., Krogdahl, S., Møller-Pedersen, B., 2009. A Collection Operator for Graph Transformation. In: Proceedings of ICMT 2009. Vol. 5563 of Lecture Notes in Computer Science. Springer, pp. 67–82.

Habel, A., Pennemann, K.-H., 2009. Correctness of High-Level Transformation Systems Relative to Nested Conditions. Mathematical Structures in Computer Science 19 (2), 245–296.

Hoffmann, B., Janssens, D., van Eetvelde, N., 2006. Cloning and Expanding Graph Transformation Rules for Refactoring. Electronic Notes in Theoretical Computer Science 152, 53–67.

Kuske, S., Gogolla, M., Kollmann, R., Kreowski, H.-J., 2002. An Integrated Semantics for UML Class, Object and State Diagrams Based on Graph Transformation. In: Proceedings of IFM 2002. Vol. 2335 of Lecture Notes in Computer Science. Springer, pp. 11–28.

Lack, S., Sobociński, P., 2005. Adhesive and Quasiadhesive Categories. Theoretical Informatics and Applications 39 (3), 511–545.

Löwe, M., 1993. Algebraic Approach to Single-Pushout Graph Transformation. Theoretical Computer Science 109, 181–224.

Reisig, W., Rozenberg, G., 1998. Lectures on Petri Nets I: Basic Models. Vol. 1491 of Lecture Notes in Computer Science. Springer.

Rensink, A., Kuperus, J.-H., 2009. Repotting the Geraniums: On Nested Graph Transformation Rules. Electronic Communications of the EASST 18, 1–15.

Rozenberg, G. (Ed.), 1997. Handbook of Graph Grammars and Computing by Graph Transformation. Vol 1: Foundations. World Scientific.

Rozenberg, G., Lindenmayer, A., 1976. Automata, Languages, and Development. North Holland.

Schürr, A., Winter, A., Zündorf, A., 1999. The PROGRES-Approach: Language and Environment. In: Handbook of Graph Grammars and Computing by Graph Transformation. Applications, Languages and Tools. Vol. 2. World Scientific, pp. 487–550.

Taentzer, G., 1996. Parallel and Distributed Graph Transformation: Formal Description and Application to Communication Based Systems. Ph.D. thesis, Technische Universität Berlin.

Taentzer, G., 2004. AGG: A Graph Transformation Environment for Modeling and Validation of Software. In: Proceedings of AGTIVE 2003. Vol. 3062 of Lecture Notes in Computer Science. Springer, pp. 446–456.

Taentzer, G., Beyer, M., 1994. Amalgamated Graph Transformations and Their Use for Specifying AGG - an Algebraic Graph Grammar System. In: Graph Transformations in Computer Science. Vol. 776 of Lecture Notes in Computer Science. Springer, pp. 380–394.

Varró, D., 2002. A Formal Semantics of UML Statecharts by Model Transition Systems. In: Proceedings of ICGT 2002. Vol. 2505 of Lecture Notes in Computer Science. Springer, pp. 378–392.

Appendix A. Proofs of Facts and Theorems

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

A.1. Proof of Thm. 4.1

First, we consider the construction without application conditions.

$$\begin{tikzcd}
L_0 \arrow[d, "s_{1,L}"'] \arrow[dr, phantom, "(1_1)"] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "s_{1,K}"] \arrow[dr, phantom, "(2_1)"] & R_0 \arrow[d, "s_{1,R}"] \\
L_1 & K_1 \arrow[l, "l_1"] \arrow[r, "r_1"'] & R_1
\end{tikzcd}
\qquad
\begin{tikzcd}
L_0 \arrow[d, "s_{1,L}"'] \arrow[dr, phantom, "(1'_1)"] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "w_1"] \arrow[dr, phantom, "(3_1)"] & R_0 \arrow[d, "e_{11}"] \\
L_1 & L_{10} \arrow[l, "u_1"] \arrow[r, "v_1"'] & E_1
\end{tikzcd}$$

Since s_1 is a kernel morphism the diagrams (1_1) and (2_1) in the right are pullbacks, and (1_1) has a pushout complement $(1'_1)$ for $s_{1,L} \circ l_0$ (see Def. 4.1). Construct the pushout (3_1). 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 the below diagram on the left. We obtain an induced morphism $s_{13} : S_1 \to R_0$ 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 Lem. B.1 implies that there is a unique morphism $l_{10} : K_1 \to L_{10}$ with $l_{10} \circ s_{1,K} = w_1$, $u_1 \circ l_{10} = l_1$, and $l_{10} \in \mathcal{M}$ depicted in the middle. Then we can construct pushouts (6_1) – (9_1) on the right as a decomposition of pushout (3_1) above which leads to $\overline{L_1}$ and $\overline{K_1}$ of the complement rule, and e_{11} and e_{12} are jointly epimorphic because $(7_1) + (9_1)$ is a pushout.

$$\begin{tikzcd}
 & K_0 \arrow[dr, "s_{11}"] \arrow[drr, "r_0"] & & \\
P_1 \arrow[ur] \arrow[dr] \arrow[r, phantom, "(5_1)"] & & S_1 \arrow[r, "s_{13}"] & R_0 \arrow[dd, "s_{1,R}"] \\
 & B_1 \arrow[ur, "s_{12}"] \arrow[urr, "b_1"'] \arrow[d] \arrow[r, phantom, "(4_1)"] & & \\
 & C_1 \arrow[rr, "c_1"'] & & R_1
\end{tikzcd}
\qquad
\begin{tikzcd}
L_0 \arrow[d, "s_{1,L}"'] \arrow[dr, phantom, "(1'_1)"] & K_0 \arrow[l, "l_0"'] \arrow[d, "w_1"] \arrow[ddr, "s_{1,K}"] & \\
L_1 & L_{10} \arrow[l, "u_1"'] & \\
 & & K_1 \arrow[ul, "l_{10}"'] \arrow[ull, "l_1"]
\end{tikzcd}
\qquad
\begin{tikzcd}
K_0 \arrow[r, "s_{11}"] \arrow[d, "s_{1,K}"'] \arrow[dr, phantom, "(6_1)"] & S_1 \arrow[r, "s_{13}"] \arrow[d, "s_{14}"] \arrow[dr, phantom, "(7_1)"] & R_0 \arrow[d, "u_{12}"] \arrow[dd, bend left, "e_{11}"] \\
K_1 \arrow[r, "v_{11}"] \arrow[d, "l_{10}"'] \arrow[dr, phantom, "(8_1)"] & \overline{K_1} \arrow[r, "\overline{w_1}"] \arrow[d, "\overline{l_1}"] \arrow[dr, phantom, "(9_1)"] & R_{10} \arrow[d, "\overline{u_1}"] \\
L_{10} \arrow[r, "u_{11}"'] & \overline{L_1} \arrow[r, "e_{12}"'] & E_1
\end{tikzcd}$$

The pushout (4_1) can be decomposed into pushouts (10_1) and (11_1) as shown on the left below obtaining 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 Lem. B.2 as shown in the middle. Now Lem. B.1 implies that there is a unique morphism $\overline{r_1} : \overline{K_1} \to \overline{R_1}$ in the right diagram with $\overline{r_1} \circ s_{14} = u_{13}$, $t_1 \circ \overline{r_1} = v_{12}$, and $\overline{r_1} \in \mathcal{M}$.

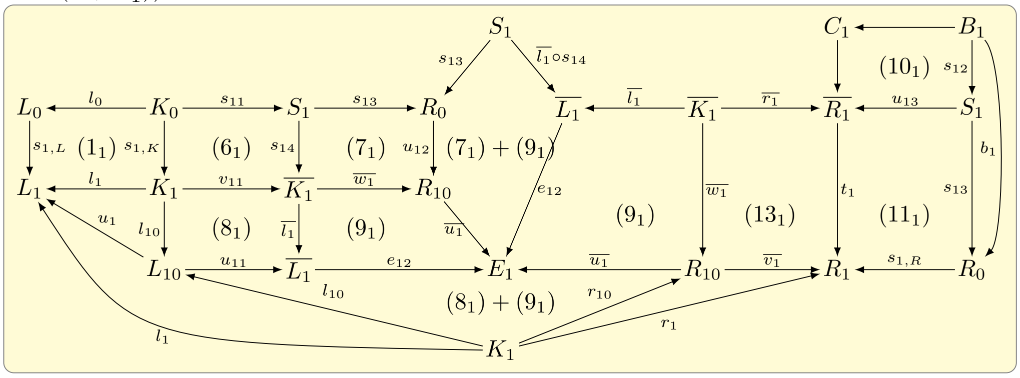
$$\begin{tikzcd}
B_1 \arrow[r, "s_{12}"] \arrow[d] \arrow[dr, phantom, "(10_1)"] & S_1 \arrow[r, "s_{13}"] \arrow[d, "u_{13}"] \arrow[dr, phantom, "(11_1)"] & R_0 \arrow[d, "s_{1,R}"] \\
C_1 \arrow[r] & \overline{R_1} \arrow[r, "t_1"'] & R_1
\end{tikzcd}
\qquad
\begin{tikzcd}
K_0 \arrow[r, "s_{11}"] \arrow[d, "s_{1,K}"'] \arrow[dr, phantom, "(6_1)"] & S_1 \arrow[r, "s_{13}"] \arrow[d, "s_{14}"] \arrow[dr, phantom, "(12_1)"] & R_0 \arrow[d, "s_{1,R}"] \\
K_1 \arrow[r, "v_{11}"'] & \overline{K_1} \arrow[r, "v_{12}"'] & R_1
\end{tikzcd}
\qquad
\begin{tikzcd}
S_1 \arrow[r, "s_{13}"] \arrow[d, "u_{13}"] \arrow[dd, bend right, "s_{14}"'] \arrow[dr, phantom, "(11_1)"] & R_0 \arrow[d, "s_{1,R}"] \\
\overline{R_1} \arrow[r, "t_1"] & R_1 \\
\overline{K_1} \arrow[u, "\overline{r_1}"] \arrow[ur, "v_{12}"'] &
\end{tikzcd}$$

The pushout (7_1) implies that there is a unique morphism $\overline{v_1}: R_{10} \to R_1$ as shown on the right, and by pushout decomposition of $(11_1) = (7_1) + (13_1)$ square (13_1) is a pushout. Moreover, $(8_1) + (9_1)$ as a pushout over $\mathcal{M}$-morphisms is also a pullback which completes the construction as shown below, leading to the required rule $\overline{p_1} = (\overline{L_1} \xleftarrow{\overline{l_1}} \overline{K_1} \xrightarrow{\overline{r_1}} \overline{R_1})$ and $p_1 = p_0 *_{E_1} \overline{p_1}$ for rules without application conditions.

$$\begin{tikzcd}
S_1 \arrow[r, "s_{13}"] \arrow[d, "s_{14}"'] \arrow[dr, phantom, "(7_1)"] & R_0 \arrow[d, "u_{12}"] \arrow[ddr, "s_{1,R}"] & \\
\overline{K_1} \arrow[r, "\overline{w_1}"] \arrow[drr, "v_{12}"'] & R_{10} \arrow[dr, "\overline{v_1}"] & \\
 & & R_1
\end{tikzcd}$$

For the application conditions, suppose $\mathrm{ac}_1 \cong \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(v_1, \mathrm{ac}_1'))$ for $p_1^* = (L_1 \xleftarrow{u_1} L_{10} \xrightarrow{v_1} E_1)$ with $v_1 = e_{12} \circ u_{11}$ and ac_1' on L_{10}. Now define $\overline{\mathrm{ac}_1} = \mathrm{Shift}(u_{11}, \mathrm{ac}_1')$, which is an application condition on $\overline{L_1}$. We have to show that $(p_1, \mathrm{ac}_{p_0 *_{E_1} \overline{p_1}}) \cong (p_1, \mathrm{ac}_1)$. By construction of the E_1-concurrent rule we have that $\mathrm{ac}_{p_0 *_{E_1} \overline{p_1}} \cong \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(e_{12}, \overline{\mathrm{ac}_1})) \cong \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(e_{12}, \mathrm{Shift}(u_{11}, \mathrm{ac}_1'))) \cong \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(e_{12} \circ u_{11}, \mathrm{ac}_1')) \cong \mathrm{Shift}(s_{1,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(v_1, \mathrm{ac}_1')) \cong \mathrm{ac}_1$.

$$\begin{tikzcd}
S_1 \arrow[r, "s_{14}"] \arrow[d, "s_{13}"'] \arrow[dr, phantom, "(7_1)"] & \overline{K_1} \arrow[r, "\overline{r_1}"] \arrow[d, "\overline{w_1}"'] \arrow[dr, phantom, "(13_1)"] & \overline{R_1} \arrow[d, "t_1"] \\
R_0 \arrow[r, "u_{12}"'] & R_{10} \arrow[r, "\overline{v_1}"'] & R_1
\end{tikzcd}$$



A.2. Proof of Fact 5.1

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

Consider the colimits $(\tilde{L}_s, (t_{i,L})_{i=0,\ldots,n})$ of $(s_{i,L})_{i=1,\ldots,n}$, $(\tilde{K}_s, (t_{i,K})_{i=0,\ldots,n})$ of $(s_{i,K})_{i=1,\ldots,n}$, and $(\tilde{R}_s, (t_{i,R})_{i=0,\ldots,n})$ of $(s_{i,R})_{i=1,\ldots,n}$, with $t_{0,*} = t_{i,*} \circ s_{i,*}$ for $* \in \{L, K, R\}$ in the right diagram. 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 $t_i \in \mathcal{M}$.

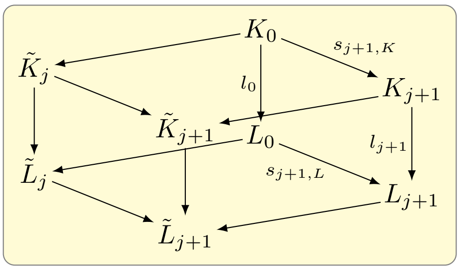
$$\begin{tikzcd}
K_0 \arrow[rr, "s_{i,K}"] \arrow[dr, "t_{0,K}"] \arrow[ddr, bend right, "t_{0,L} \circ l_0"'] & & K_i \arrow[dl, "t_{i,K}"'] \arrow[ddl, bend left, "t_{i,L} \circ l_i"] \\
 & \tilde{K}_s \arrow[d, "\tilde{l}_s"] & \\
 & \tilde{L}_s &
\end{tikzcd}
\qquad
\begin{tikzcd}
K_0 \arrow[rr, "s_{i,K}"] \arrow[dr, "t_{0,K}"] \arrow[ddr, bend right, "t_{0,R} \circ r_0"'] & & K_i \arrow[dl, "t_{i,K}"'] \arrow[ddl, bend left, "t_{i,R} \circ r_i"] \\
 & \tilde{K}_s \arrow[d, "\tilde{r}_s"] & \\
 & \tilde{R}_s &
\end{tikzcd}$$

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

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

$$\begin{tikzcd}
K_i \arrow[r] \arrow[d, "l_i"'] \arrow[dr, phantom, "(16_{ij})"] & \tilde{K}_j \arrow[d] & K_0 \arrow[r, "s_{1,K}"] \arrow[d, "l_0"'] \arrow[dr, phantom, "(1_1)"] & K_1 \arrow[r] \arrow[d, "l_1"'] \arrow[dr, phantom, "(16_{11})"] & \tilde{K}_1 \arrow[d] \\
L_i \arrow[r] & \tilde{L}_j & L_0 \arrow[r, "s_{1,L}"'] & L_1 \arrow[r] & \tilde{L}_1
\end{tikzcd}$$

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



Induction step $j \to j+1$: Construct $\tilde{L}_{j+1} = \tilde{L}_j +_{L_0} L_{j+1}$ and $\tilde{K}_{j+1} = \tilde{K}_j +_{K_0} K_{j+1}$ as pushouts in the right cube, and we have that the top and bottom faces as pushouts, the back faces as pullbacks, and by the van Kampen property also the front faces are pullbacks. Moreover, by Lem. B.3 the front faces have the pushout complement property, and by Lem. B.4 this holds also for (16_{0j}) and (16_{ij}) as compositions.

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

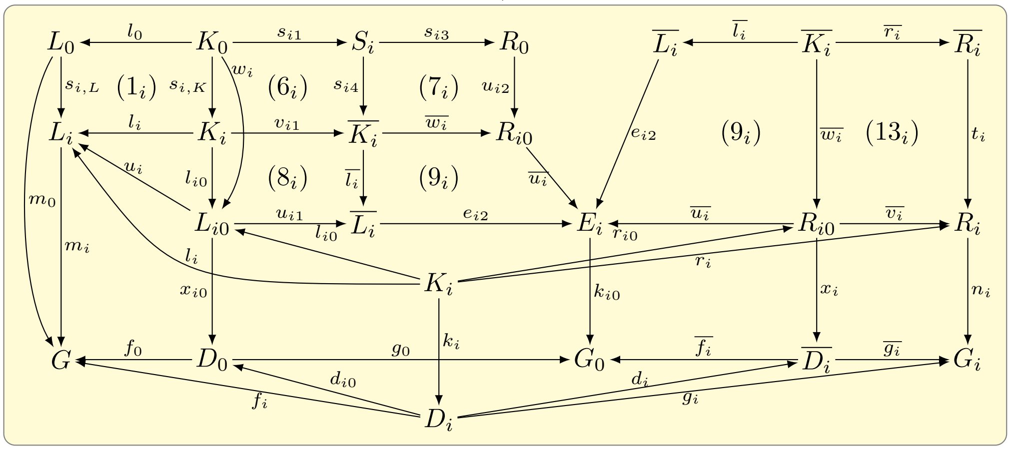
$$\begin{tikzcd}
p_0 : & \mathrm{ac}_0 \rhd & L_0 \arrow[d, "s_{i,L}"'] \arrow[dr, phantom, "(1'_i)"] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "w_i"'] \arrow[dr, phantom, "(3_i)"] & R_0 \arrow[d, "e_{i1}"'] \\
p_i^* : & \mathrm{ac}_i \rhd & L_i \arrow[d, "t_{i,L}"'] \arrow[dr, phantom, "(17_i)"] & L_{i0} \arrow[l, "u_i"'] \arrow[r, "v_i"] \arrow[d, "\tilde{l}_i"'] \arrow[dr, phantom, "(18_i)"] & E_i \arrow[d, "\tilde{k}_i"'] \\
\tilde{p}_s^* : & \tilde{\mathrm{ac}}_s \rhd & \tilde{L}_s & \tilde{L}_0 \arrow[l, "\tilde{u}"'] \arrow[r, "\tilde{v}"] & \tilde{E}
\end{tikzcd}$$

Moreover, we have pushout complements (17_i) resp. $(17_i) + (1'_i)$ for $t_{i,L} \circ l_i$ as shown on the right. Since ac_0 and ac_i are complement-compatible for all i we have that $\mathrm{ac}_i \cong \mathrm{Shift}(s_{i,L}, \mathrm{ac}_0) \wedge \mathrm{L}(p_i^*, \mathrm{Shift}(v_i, \mathrm{ac}'_i))$. For any ac'_i, it holds that $\mathrm{Shift}(t_{i,L}, \mathrm{L}(p_i^*, \mathrm{Shift}(v_i, \mathrm{ac}'_i))) \cong \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{k}_i \circ v_i, \mathrm{ac}'_i)) \cong \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{v}, \mathrm{Shift}(\tilde{l}_i, \mathrm{ac}'_i)))$, since all squares are pushouts by pushout-pullback decomposition and the uniqueness of pushout complements. Define $\mathrm{ac}_i^* := \mathrm{Shift}(\tilde{l}_i, \mathrm{ac}'_i)$ as an application condition on $\tilde{L}_0$. It follows that $\tilde{\mathrm{ac}}_s = \bigwedge_{i=1,\dots,n} \mathrm{Shift}(t_{i,L}, \mathrm{ac}_i) \cong \bigwedge_{i=1,\dots,n} (\mathrm{Shift}(t_{i,L} \circ s_{i,L}, \mathrm{ac}_0) \wedge \mathrm{Shift}(t_{i,L}, \mathrm{L}(p_i^*, \mathrm{Shift}(v_i, \mathrm{ac}'_i)))) \cong \mathrm{Shift}(t_{0,L}, \mathrm{ac}_0) \wedge \bigwedge_{i=1,\dots,n} \mathrm{L}(\tilde{p}_s^*, \mathrm{Shift}(\tilde{v}, \mathrm{ac}_i^*))$.

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

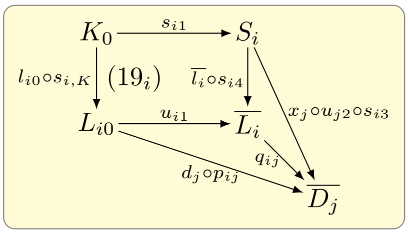
A.3. Proof of Fact 5.3

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



We have to show the pairwise parallel independence. From the constructions of the complement rule and the Concurrency Theorem we obtain the above diagram for all $i = 1, \ldots, n$. For $i \neq j$, from weakly independent matches it follows that we have a morphism $p_{ij} : L_{i0} \to D_j$ with $f_j \circ p_{ij} = m_i \circ u_i$. It follows that $f_j \circ p_{ij} \circ w_i = m_i \circ u_i \circ w_i = m_i \circ s_{i,L} \circ l_0 = m_0 \circ l_0 = m_j \circ s_{j,L} \circ l_0 = m_j \circ u_j \circ w_j = m_j \circ u_j \circ l_{j0} \circ s_{j,K} = m_j \circ l_j \circ s_{j,K} = f_j \circ k_j \circ s_{j,K}$, and with $f_j \in \mathcal{M}$ we have that $p_{ij} \circ w_i = k_j \circ s_{jk}$ $(*)$.

Now consider the pushout $(19_i) = (6_i) + (8_i)$ in comparison with object $\overline{D_j}$ and morphisms $d_j \circ p_{ij}$ and $x_j \circ u_{j2} \circ s_{i3}$ as shown below. We have that $d_j \circ p_{ij} \circ l_{i0} \circ s_{i,K} = d_j \circ p_{ij} \circ w_i \overset{(*)}{=} d_j \circ k_j \circ s_{j,K} = x_j \circ r_{j0} \circ s_{j,K} = x_j \circ \overline{w_j} \circ v_{j1} \circ s_{j,K} = x_j \circ u_{j2} \circ s_{j3} \circ s_{j1} = x_j \circ u_{j2} \circ r_0 = x_j \circ u_{j2} \circ s_{i3} \circ s_{i1}$. Now pushout (19_i) induces a unique morphism q_{ij} with $q_{ij} \circ u_{i1} = d_j \circ p_{ij}$ and $q_{ij} \circ \overline{l_i} \circ s_{i4} = x_j \circ u_{j2} \circ s_{i3}$.



For the parallel independence of $G_0 \xRightarrow{\overline{p_i}, \overline{m_i}} G_i$, $G_0 \xRightarrow{\overline{p_j}, \overline{m_j}} G_j$, we have to show that $q_{ij} : \overline{L_i} \to \overline{D_j}$ satisfies $\overline{f_j} \circ q_{ij} = k_{i0} \circ e_{i2} =: \overline{m_i}$.

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 $d_{j0} \circ p_{ij} = x_{i0}$ $(**)$. This means that $\overline{f_j} \circ q_{ij} \circ u_{i1} = \overline{f_j} \circ d_j \circ p_{ij} = g_0 \circ d_0 \circ p_{ij} \overset{(**)}{=} g_0 \circ x_{i0} = k_{i0} \circ e_{i2} \circ u_{i1}$. In addition, we have that $\overline{f_j} \circ q_{ij} \circ \overline{l_i} \circ s_{i4} = \overline{f_j} \circ x_j \circ u_{j2} \circ s_{i3} = k_{j0} \circ \overline{u_j} \circ u_{j2} \circ s_{i3} = k_{i0} \circ \overline{u_i} \circ u_{i2} \circ s_{i3} = k_{i0} \circ e_{i2} \circ \overline{l_i} \circ s_{i4}$. Since (19_i) is a pushout, u_{i1} and $\overline{l_i} \circ s_{i4}$ are jointly epimorphic, and it follows that $\overline{f_j} \circ q_{ij} = k_{i0} \circ e_{i2}$.

If ac_0 and ac_i are not complement-compatible, then $\overline{\mathrm{ac}_i} = \mathrm{true}$ and trivially $\overline{g_j} \circ q_{ij} \models \overline{\mathrm{ac}_i}$ for all $j \neq i$. Otherwise, we have that $g_j \circ p_{ij} \models \mathrm{ac}_i'$, 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 $\overline{g_j} \circ q_{ij} \circ u_{i1} \models \mathrm{ac}_i'$, which is equivalent to $\overline{g_j} \circ q_{ij} \models \mathrm{Shift}(u_{i1}, \mathrm{ac}_1') = \overline{ac_i}$.

A.4. Proof of Thm. 5.1

1. Synthesis. We have to show that $\tilde{p}_s$ is applicable to G leading to an amalgamated transformation $G \xRightarrow{\tilde{p}_s,\tilde{m}} H$ with $m_i = \tilde{m} \circ t_{i,L}$, where $t_i : p_i \to \tilde{p}_i$ is the kernel morphism constructed in Fact 5.1.

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

$$\begin{tikzcd}
\mathrm{ac}_0 \triangleright L_0 \arrow[d, "s_{i,L}"'] \arrow[dr, phantom, "(1_i)"] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "s_{i,K}"] \arrow[dr, phantom, "(2_i)"] & R_0 \arrow[d, "s_{i,R}"] \\
\mathrm{ac}_i \triangleright L_i \arrow[d, "t_{i,L}"'] \arrow[dr, phantom, "(14_i)"] & K_i \arrow[l, "l_i"] \arrow[r, "r_i"'] \arrow[d, "t_{i,K}"] \arrow[dr, phantom, "(15_i)"] & R_i \arrow[d, "t_{i,R}"] \\
\tilde{\mathrm{ac}}_s \triangleright \tilde{L}_s & \tilde{K}_s \arrow[l, "\tilde{l}_s"] \arrow[r, "\tilde{r}_s"'] & \tilde{R}_s
\end{tikzcd}$$

$$\begin{tikzcd}
L_i \arrow[d, "m_i"'] \arrow[dr, phantom, "(20_i)"] & K_i \arrow[l, "l_i"'] \arrow[r, "r_i"] \arrow[d, "k_i"] \arrow[dr, phantom, "(21_i)"] & R_i \arrow[d, "n_i"] \\
G & D_i \arrow[l, "f_i"] \arrow[r, "g_i"'] & G_i
\end{tikzcd}$$

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

Using Fact 5.3, we know that we can apply p_0 via m_0 leading to a direct transformation $G \xRightarrow{p_0,m_0} G_0$ given by pushouts (20_0) and (21_0) below. Moreover, we find decompositions of pushouts (20_0) and (20_i) into pushouts $(1'_i)$ and (22_i) resp. (22_i) and (23_i) by $\mathcal{M}$-pushout pullback decomposition and uniqueness of pushout complements as shown in the bottom diagram below.

$$\begin{tikzcd}
L_0 \arrow[d, "m_0"'] \arrow[dr, phantom, "(20_0)"] & K_0 \arrow[l, "l_0"'] \arrow[r, "r_0"] \arrow[d, "k_0"] \arrow[dr, phantom, "(21_0)"] & R_0 \arrow[d, "n_0"] \\
G & D_0 \arrow[l, "f_0"] \arrow[r, "g_0"'] & G_0
\end{tikzcd}$$

$$\begin{tikzcd}
L_0 \arrow[d, "s_{i,L}"'] \arrow[dr, phantom, "(1'_i)"] & K_0 \arrow[l, "l_0"'] \arrow[d, "w_i"] \arrow[dr, "s_{i,K}"] & \\
L_i \arrow[d, "m_i"'] \arrow[dr, phantom, "(22_i)"] & L_{i0} \arrow[l, "u_i"] \arrow[d, "x_{i0}"] \arrow[dr, phantom, "(23_i)"] & K_i \arrow[l, "l_{i0}"] \arrow[d, "k_i"] \\
G & D_0 \arrow[l, "f_0"] & D_i \arrow[l, "d_{i0}"]
\end{tikzcd}$$

Since we have consistent matches, $m_i \circ s_{i,L} = m_0$ for all $i = 1, \ldots, n$. Then the colimit $\tilde{L}_s$ implies that there is a unique morphism $\tilde{m} : \tilde{L}_s \to G$ with $\tilde{m} \circ t_{i,L} = m_i$ and $\tilde{m} \circ t_{0,L} = m_0$ (see (a) below). Moreover, $m_i \models \mathrm{ac}_i \Rightarrow \tilde{m} \circ t_{i,L} \models \mathrm{ac}_i \Rightarrow \tilde{m} \models \mathrm{Shift}(t_{i,L}, \mathrm{ac}_i)$ for all $i = 1, \ldots, n$, and thus $\tilde{m} \models \tilde{\mathrm{ac}}_s = \bigwedge_{i=1,\ldots,n} \mathrm{Shift}(t_{i,L}, \mathrm{ac}_i)$

Weakly independent matches means that there exist morphisms p_{ij} with $f_j \circ p_{ij} = m_i \circ u_i$ for $i \neq j$. Construct D as the limit of $(d_{i0})_{i=1,\ldots,n}$ 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 $d_{i0} \circ p_{ji} = x_{j0}$. It follows that $d_{i0} \circ p_{ji} \circ l_{j0} = x_{j0} \circ l_{j0}$, and together with $d_{i0} \circ k_i = x_{i0} \circ l_{i0}$ limit D implies that there exists a unique morphism r_j with $d_i \circ r_j = p_{ji} \circ l_{j0}$, $d_i \circ r_i = k_i$, and $d_0 \circ r_j = x_{j0} \circ l_{j0}$ (see (b)).

(a)
$$\begin{tikzcd}
L_0 \arrow[rr, "s_{i,L}"] \arrow[dr, "t_{0,L}"] \arrow[ddr, "m_0"'] & & L_i \arrow[dl, "t_{i,L}"'] \arrow[ddl, "m_i"] \\
& \tilde{L}_s \arrow[d, "\tilde{m}"] & \\
& G &
\end{tikzcd}$$

(b)
$$\begin{tikzcd}
& K_j \arrow[d, "r_j"] \arrow[ddl, "{i\neq j: p_{ji}\circ l_{j0}} \\ {i=j: k_i}"'] \arrow[ddr, "x_{j0}\circ l_{j0}"] & \\
& D \arrow[dl, "d_i"'] \arrow[dr, "d_0"] & \\
D_i \arrow[rr, "d_{i0}"'] & & D_0
\end{tikzcd}$$

(c)
$$\begin{tikzcd}
K_0 \arrow[rr, "s_{i,K}"] \arrow[dr, "t_{0,K}"] \arrow[ddr, "k_j\circ s_{j,K}"'] & & K_i \arrow[dl, "t_{i,K}"'] \arrow[ddl, "{i\neq j: p_{ij}\circ l_{i0}} \\ {i=j: k_i}"] \\
& \tilde{K}_s \arrow[d, "\tilde{r}_j"] & \\
& D_j &
\end{tikzcd}$$

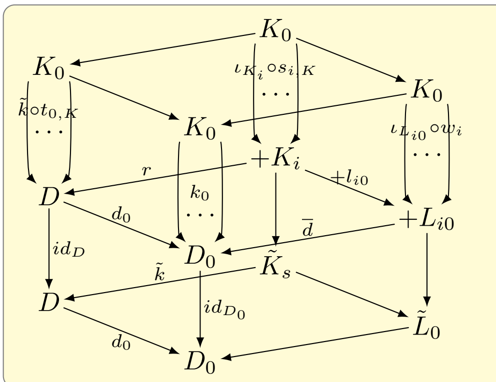
(d)
$$\begin{tikzcd}
& \tilde{K}_s \arrow[d, "\tilde{k}"] \arrow[ddl, "\tilde{r}_i"'] \arrow[ddr, "\tilde{r}"] & \\
& D \arrow[dl, "d_i"'] \arrow[dr, "d_0"] & \\
D_i \arrow[rr, "d_{i0}"'] & & D_0
\end{tikzcd}$$

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 $\tilde{r}_j \circ t_{i,K} = p_{ij} \circ l_{i0}$, $\tilde{r}_j \circ t_{j,K} = k_j$, and $\tilde{r}_j \circ t_{0,K} = k_j \circ s_{j,K}$ (see (c) above). Since $d_{i0} \circ \tilde{r}_i \circ t_{i,K} = d_{i0} \circ k_i = q_i \circ l_{i0} = d_{j0} \circ p_{ij} \circ l_{i0} = d_{j0} \circ \tilde{r}_j \circ t_{i,K}$ 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 that $d_{i0} \circ \tilde{r}_i = d_{j0} \circ \tilde{r}_j =: \tilde{r}$. From limit D it now follows that there exists a unique morphism $\tilde{k}$ with $d_i \circ \tilde{k} = \tilde{r}_i$ and $d_0 \circ \tilde{k} = \tilde{r}$ (see (d)).

$$\begin{tikzcd}
\tilde{L}_s \arrow[d, "\tilde{m}"'] \arrow[dr, phantom, "(20_s)"] & \tilde{K}_s \arrow[l, "\tilde{l}_s"'] \arrow[d, "\tilde{k}"] & K_i \arrow[r, "r_i"] \arrow[d, "l_{i0}"'] \arrow[dr, phantom, "(24_i)"] & D \arrow[r, "d_i"] \arrow[d, "x_i"] \arrow[dr, phantom, "(25_i)"] & D_i \arrow[d, "d_{i0}"] & +K_i \arrow[r, "+l_{i0}"] \arrow[d, "r"'] \arrow[dr, phantom, "(25)"] & +L_{i0} \arrow[d, "\overline{d}"] \\
G & D \arrow[l, "f"] & L_{i0} \arrow[r, "x_{i0}"'] & P_i \arrow[r, "y_{i0}"'] & D_0 & D \arrow[r, "d_0"'] & D_0
\end{tikzcd}$$

We have to show that (20_s) in the left of the right diagram with $f = f_0 \circ d_0$ is a pushout. With $f \circ \tilde{k} \circ t_{i,K} = f_0 \circ d_0 \circ \tilde{k} \circ t_{i,K} = f_0 \circ \tilde{r} \circ t_{i,K} = f_0 \circ d_{i0} \circ \tilde{r}_i \circ t_{i,K} = f_0 \circ d_{i0} \circ k_i = f_i \circ k_i = m_i \circ l_i = \tilde{m} \circ t_{i,L} \circ l_i = \tilde{m} \circ \tilde{l}_s \circ t_{i,K}$ and $f \circ \tilde{k} \circ t_{0,K} = f_0 \circ d_0 \circ \tilde{k} \circ t_{0,K} = f_0 \circ \tilde{r} \circ t_{0,K} = f_0 \circ d_{i0} \circ \tilde{r}_i \circ t_{0,K} = f_0 \circ d_{i0} \circ k_i \circ s_{i,K} = f_0 \circ k_0 = m_0 \circ l_0 = \tilde{m} \circ t_{0,L} \circ l_0 = \tilde{m} \circ \tilde{l}_s \circ t_{0,K}$ and $\tilde{K}_s$ being colimit it follows that $f \circ \tilde{k} = \tilde{m} \circ \tilde{l}_s$, thus the square commutes.

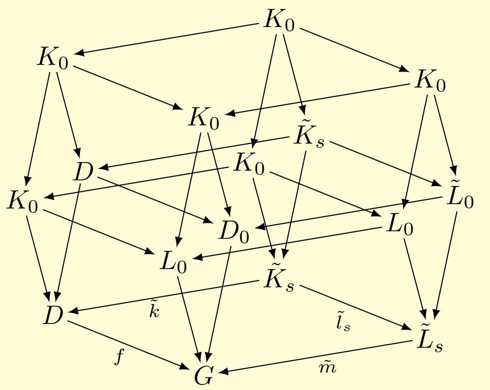
Pushout (23_i) can be decomposed into pushouts (24_i) and (25_i) in the middle of the diagram above. Using Lem. B.5 it follows that D_0 is the colimit of $(x_i)_{i=1,\ldots,n}$, because (23_i) is a pushout, D is the limit of $(d_{i0})_{i=1,\ldots,n}$, and we have morphisms p_{ij} with $d_{j0} \circ p_{ij} = q_i$. Then Lem. B.6 implies that also (25) on the right is a pushout, where $+$ represents the coproduct construction with index $i = 1, \ldots, n$ with injections ι_{K_i} and $\iota_{L_{i0}}$, respectively.



$$\begin{tikzcd}
\tilde{K}_s \arrow[r] \arrow[d, "\tilde{k}"'] \arrow[dr, phantom, "(26)"] & \tilde{L}_0 \arrow[d] \\
D \arrow[r, "d_0"'] & D_0
\end{tikzcd}$$

Consider the n-ary coequalizers $\tilde{K}_s$ of $(\iota_{K_i} \circ s_{i,K} : K_0 \to +K_i)_{i=1,\ldots,n}$ (which is actually $\tilde{K}_s$ by construction of colimits), $\tilde{L}_0$ of $(iota_{L_{i0}} \circ w_i : K_0 \to +L_{i0})_{i=1,\ldots,n}$ (as already constructed in Fact 5.1), D of $(\tilde{k} \circ t_{0,K} : K_0 \to D)_{i=1,\ldots,n}$, and D_0 of $(k_0 : K_0 \to D_0)_{i=1,\ldots,n}$. In the right cube, the top square with identical morphisms is a pushout, the top cube commutes, and the middle square is pushout (25) from above. Using Lem. B.7 it follows that also the bottom square (26) constructed of the four coequalizers is a pushout.

Now consider the cube below, where the top and middle squares are pushouts and the two top cubes commute. Using again Lem. B.7 it follows that (20_s) in the bottom is actually a pushout, where $(27) = (1'_i) + (17_i)$ is a pushout by composition. Now we can construct pushout (21_s) which completes the direct transformation $G \overset{\tilde{p}_s, \tilde{m}}{\Longrightarrow} H$.



$$\begin{tikzcd}
K_0 \arrow[r] \arrow[d, "l_0"'] \arrow[dr, phantom, "(27)"] & \tilde{L}_0 \arrow[d] \\
L_0 \arrow[r, "t_{0,K}"'] & \tilde{L}_s
\end{tikzcd}$$

$$\begin{tikzcd}
\tilde{L}_s \arrow[d, "\tilde{m}"'] \arrow[dr, phantom, "(20_s)"] & \tilde{K}_s \arrow[l, "\tilde{l}_s"'] \arrow[r, "\tilde{r}_s"] \arrow[d, "\tilde{k}"] \arrow[dr, phantom, "(21_s)"] & \tilde{R}_s \arrow[d, "\tilde{n}"] \\
G & D \arrow[l, "f"] \arrow[r, "g"'] & H
\end{tikzcd}$$

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 4.1 with $m_i = \tilde{m} \circ t_{i,L}$. We have to show that this bundle of transformation is s-amalgamable. Applying again Fact 4.1 we obtain transformations $G \xRightarrow{p_0,m_0^i} G_0^i \xRightarrow{\overline{p_i}} G_i$ with $m_0^i = m_i \circ s_{i,L}$. It follows that $m_0^i = m_i \circ s_{i,L} = \tilde{m} \circ t_{i,L} \circ s_{i,L} = \tilde{m} \circ t_{0,L} = \tilde{m} \circ t_{j,L} \circ s_{j,L} = m_j \circ s_{j,L}$ and thus we have consistent matches with $m_0 := m_0^i$ well-defined and $G_0 = G_0^i$.

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

$$\begin{tikzcd}
 & K_0 \arrow[r, "l_0"] \arrow[d] \arrow[dr, phantom, "(27)"] & L_0 \arrow[d, "t_{0,L}"] \\
\tilde{K}_s \arrow[r] \arrow[d, "\tilde{k}"'] \arrow[dr, phantom, "(26)"] & \tilde{L}_0 \arrow[r, "\tilde{u}"] \arrow[d] \arrow[dr, phantom, "(28)"] & \tilde{L}_s \arrow[d, "\tilde{m}"] \\
D \arrow[r, "d_0"'] & D_0 \arrow[r, "f_0"'] & G
\end{tikzcd}$$

Now Lem. B.6 implies that there is a decomposition into pushouts (24_i) with colimit D_0 of $(x_i)_{i=1,\ldots,n}$ and pushout (25_i) by $\mathcal{M}$-pushout pullback decomposition depicted below. Since D_0 is the colimit of $(x_i)_{i=1,\ldots,n}$ and (25_j) is a pushout it follows that D_j is the colimit of $(x_i)_{i=1,\ldots,j-1,j+1,\ldots,n}$ with morphisms $q_{ij} : P_i \to D_j$ and $d_{j0} \circ q_{ij} = y_{i0}$. Thus we obtain for all $i \neq j$ a morphism $p_{ij} = q_{ij} \circ x_{i0}$ and $f_j \circ p_{ij} = f_0 \circ d_{j0} \circ q_{ij} \circ x_{i0} = f_0 \circ y_{i0} \circ x_{i0} = m_i \circ u_i$.

$$\begin{tikzcd}
 & K_i \arrow[r, "r_i"] \arrow[d, "l_{i0}"'] \arrow[dr, phantom, "(24_i)"] & D \arrow[r, "d_i"] \arrow[d, "x_i"] \arrow[dr, phantom, "(25_i)"] & D_i \arrow[d, "d_{i0}"] \\
K_0 \arrow[r, "w_i"] \arrow[d, "l_0"'] \arrow[dr, phantom, "(1'_i)"] & L_{i0} \arrow[r, "x_{i0}"'] \arrow[d, "u_i"] & P_i \arrow[r, "y_{i0}"'] & D_0 \arrow[d, "f_0"] \\
L_0 \arrow[r, "s_{i,L}"'] & L_i \arrow[rr, "m_i"'] & & G
\end{tikzcd}$$

$$\begin{tikzcd}
 & & L_{i0} \arrow[d, "x_{i0}"'] \\
 & D \arrow[r, "x_i"] \arrow[dl, "x_j"'] \arrow[dr, "d_j"] & P_i \arrow[d, "q_{ij}"'] \arrow[ddl, bend left, "y_{i0}"] \\
P_j \arrow[dr, "y_{j0}"'] \arrow[rr, phantom, "(25_j)"] & & D_j \arrow[dl, "d_{j0}"] \\
 & D_0 &
\end{tikzcd}$$

3. Bijective Correspondence. Because of the uniqueness of the used constructions, the above constructions are inverse to each other up to isomorphism.

Appendix B. Additional Lemmas

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

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

$$\begin{tikzcd}
A \arrow[r, "m"] \arrow[d, "f"'] \arrow[dr, phantom, "(1)"] & B \arrow[d, "g"] \\
C \arrow[r, "n"'] & D
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "m"] \arrow[d, "f'"] \arrow[dd, bend right, "f"'] \arrow[dr, phantom, "(2)"] & B \arrow[d, "g"] \\
C' \arrow[r, "n'"] \arrow[d, "c"] & D \\
C \arrow[ur, "n"'] &
\end{tikzcd}$$

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

$$\begin{tikzcd}
A \arrow[dr, "f^*"] \arrow[drr, "f'"] \arrow[ddr, "f"'] & & \\
 & C'' \arrow[r, "v"] \arrow[d, "v'"] \arrow[dr, phantom, "(3)"] & C' \arrow[d, "n'"] \\
 & C \arrow[r, "n"'] & D
\end{tikzcd}
\qquad
\begin{tikzcd}
 & A \arrow[dl, "id_A"'] \arrow[rr, "f^*"] \arrow[dd, "id_A"] & & C'' \arrow[dl, "v"] \arrow[dd, "v'"] \\
A \arrow[rr, "f'"] \arrow[dd, "m"'] & & C' \arrow[dd, "n'"] & \\
 & A \arrow[dl, "m"'] \arrow[rr, "f"] & & C \arrow[dl, "n"] \\
B \arrow[rr, "g"'] & & D &
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "f^*"] \arrow[d, "id_A"'] \arrow[dr, phantom, "(5)"] & C'' \arrow[d, "v"] \\
A \arrow[r, "f'"'] & C'
\end{tikzcd}$$

(4)

□

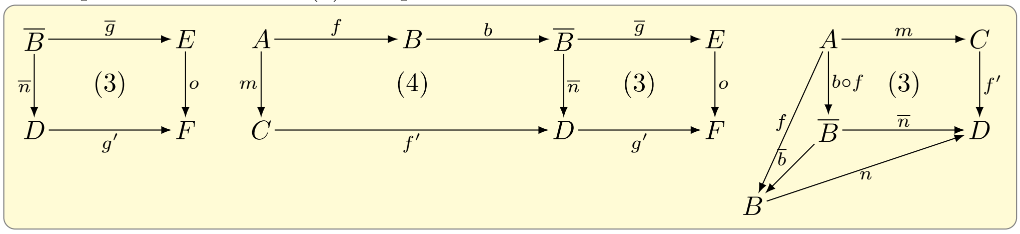
Lemma B.2 ($\mathcal{M}$ pullback-pushout decomposition). If (1) + (2) is a pullback, (1) is a pushout, (2) commutes, and $o \in \mathcal{M}$ then also (2) is a pullback.

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "m"'] \arrow[dr, phantom, "(1)"] & B \arrow[r, "g"] \arrow[d, "n"] \arrow[dr, phantom, "(2)"] & E \arrow[d, "o"] \\
C \arrow[r, "f'"'] & D \arrow[r, "g'"'] & F
\end{tikzcd}$$

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

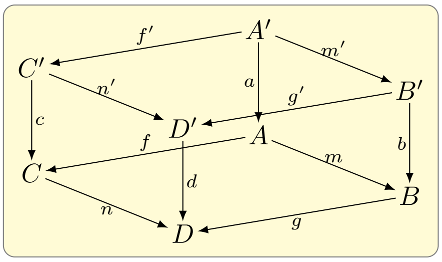
By pullback decomposition, also (4) is a pullback and we can apply Lem. B.1 with pushout (1) and $\overline{n} \in \mathcal{M}$ to obtain a unique morphism $\overline{b} \in \mathcal{M}$ with $n \circ \overline{b} = \overline{n}$ and $\overline{b} \circ b \circ f = f$. Now $n \in \mathcal{M}$ and $n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

$\overline{n} \in \mathcal{M}$ and $\overline{n} \circ b \circ \overline{b} = n \circ \overline{b} = \overline{n}$ implies that $b \circ \overline{b} = id_{\overline{B}}$, which means that B and $\overline{B}$ are isomorphic such that also (2) is a pullback.



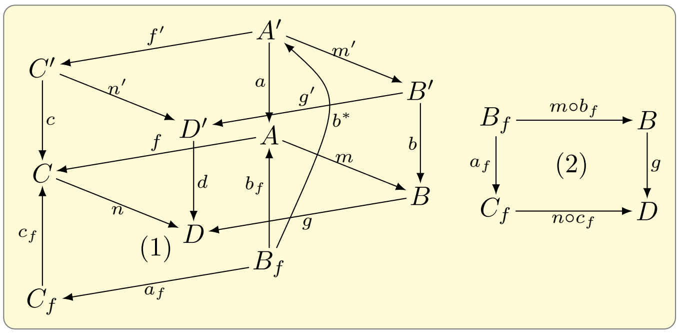
□

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



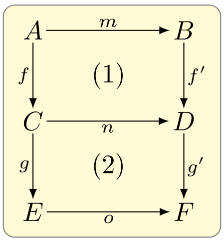
Proof. Construct the initial pushout (1) over f.

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



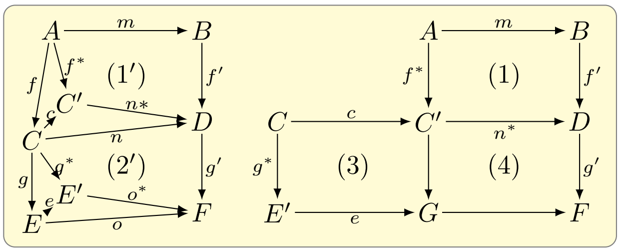
□

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



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

Now $(2')$ can be decomposed into pushouts (3) and (4), and $(1') + (4)$ is also a pushout and the pushout complement of $(g' \circ f') \circ m$. □



Lemma B.5. Given the pushouts (1_i) and (3_i) with $b_i \in \mathcal{M}$ for $i = 1, \ldots, n$, morphisms $f_{ij} : B_i \to C_j$ with $c_j \circ f_{ij} = d_i$ for all $i \neq j$, and the limit (2) of $(c_j)_{j=1,\ldots,n}$ such that g_i is the induced morphism into E with $e_i \circ g_i = a_i$ and $e_j \circ g_i = f_{ij} \circ b_i$ using $c_j \circ f_{ij} \circ b_i = d_i \circ b_i = c_i \circ a_i$, then (4) is the colimit of $(h_i)_{i=1,\ldots,n}$, where l_i is the induced morphism from pushout (3_i) compared with $\overline{e} \circ g_i = c_i \circ e_i \circ g_i = c_i \circ a_i = d_i \circ b_i$.

$$\begin{tikzcd} A_i \arrow[r, "a_i"] \arrow[d, "b_i"'] \arrow[dr, phantom, "(1_i)"] & C_i \arrow[d, "c_i"] \\ B_i \arrow[r, "d_i"'] & D \end{tikzcd} \qquad \begin{tikzcd} & E \arrow[dl, "e_j"'] \arrow[dr, "\overline{e}"] \arrow[d, phantom, "(2)"] & \\ C_j \arrow[rr, "c_j"'] & & D \end{tikzcd} \qquad \begin{tikzcd} B_i \arrow[r, "f_{ij}"] \arrow[dr, "d_i"'] & C_j \arrow[d, "c_j"] \\ & D \end{tikzcd}$$

$$\begin{tikzcd} A_i \arrow[r, "g_i"] \arrow[d, "b_i"'] \arrow[dr, phantom, "(3_i)"] & E \arrow[d, "h_i"] \arrow[ddr, "\overline{e}"] & \\ B_i \arrow[r, "k_i"] \arrow[drr, "d_i"'] & F_i \arrow[dr, "l_i"] & \\ & & D \end{tikzcd} \qquad \begin{tikzcd} & A_i \arrow[d, "g_i"] \arrow[ddl, "{\substack{i\neq j: f_{ij}\circ b_i \\ i=j: a_i}}"'] \arrow[ddr, "c_i \circ a_i"] & \\ & E \arrow[dl, "e_j"'] \arrow[dr, "\overline{e}"] \arrow[d, phantom, "(2)"] & \\ C_j \arrow[rr, "c_j"'] & & D \end{tikzcd} \qquad \begin{tikzcd} E \arrow[rr, "h_i"] \arrow[dr, "\overline{e}"'] & \arrow[d, phantom, "(4)"] & F_i \arrow[dl, "l_i"] \\ & D & \end{tikzcd}$$

Proof. We prove this by induction over n.

I.B. $n = 1$: For $n = 1$, we have that C_1 is the limit of c_1, i.e. $E = C_1$, it follows that $F_1 = C_1$ for the pushout $(3_1) = (1_1)$, and obviously (4_1) is a colimit.

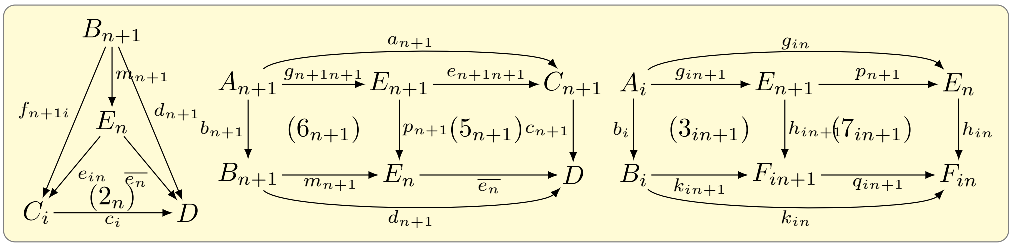
$$\begin{tikzcd} A_1 \arrow[r, "a_i"] \arrow[d, "b_i"'] \arrow[dr, phantom, "(1_1)"] & C_1 \arrow[d, "c_i"] \\ B_1 \arrow[r, "d_i"'] & D \end{tikzcd} \qquad \begin{tikzcd} & C_1 \arrow[dl, "e_i"'] \arrow[dr, "\overline{e}"] \arrow[d, phantom, "(2)"] & \\ C_1 \arrow[rr, "c_i"'] & & D \end{tikzcd} \qquad \begin{tikzcd} C_1 \arrow[rr, "h_i"] \arrow[dr, "\overline{e}"'] & \arrow[d, phantom, "(4_1)"] & D \arrow[dl, "l_i"] \\ & D & \end{tikzcd}$$

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

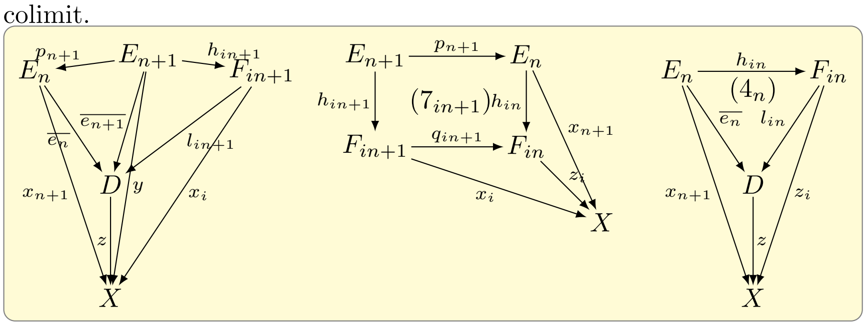
$$\begin{tikzcd} A_i \arrow[r, "a_i"] \arrow[d, "b_i"'] \arrow[dr, phantom, "(1_i)"] & C_i \arrow[d, "c_i"] \\ B_i \arrow[r, "d_i"'] & D \end{tikzcd} \qquad \begin{tikzcd} & E_n \arrow[dl, "e_{in}"'] \arrow[dr, "\overline{e_n}"] \arrow[d, phantom, "(2_n)"] & \\ C_i \arrow[rr, "c_i"'] & & D \end{tikzcd} \qquad \begin{tikzcd} A_i \arrow[r, "g_{in}"] \arrow[d, "b_i"'] \arrow[dr, phantom, "(3_{in})"] & E_n \arrow[d, "h_{in}"] \\ B_i \arrow[r, "k_{in}"] & F_{in} \end{tikzcd} \qquad \begin{tikzcd} E_n \arrow[rr, "h_{in}"] \arrow[dr, "\overline{e_n}"'] & \arrow[d, phantom, "(4_n)"] & F_{in} \arrow[dl, "l_{in}"] \\ & D & \end{tikzcd}$$

$$\begin{tikzcd} E_{n+1} \arrow[r, "e_{n+1n+1}"] \arrow[d, "p_{n+1}"'] \arrow[dr, phantom, "(5_{n+1})"] & C_{n+1} \arrow[d, "c_{n+1}"] \\ E_n \arrow[r, "\overline{e_n}"'] & D \end{tikzcd} \qquad \begin{tikzcd} & E_{n+1} \arrow[dl, "e_{in+1}"'] \arrow[dr, "\overline{e_n+1}"] \arrow[d, phantom, "(2_{n+1})"] & \\ C_i \arrow[rr, "c_i"'] & & D \end{tikzcd} \qquad \begin{tikzcd} A_i \arrow[r, "g_{in+1}"] \arrow[d, "b_i"'] \arrow[dr, phantom, "(3_{in+1})"] & E_{n+1} \arrow[d, "h_{in+1}"] \\ B_i \arrow[r, "k_{in+1}"] & F_{in+1} \end{tikzcd} \qquad \begin{tikzcd} E_{n+1} \arrow[rr, "h_{in+1}"] \arrow[dr, "\overline{e_n+1}"'] & \arrow[d, phantom, "(4_{n+1})"] & F_{in+1} \arrow[dl, "l_{in+1}"] \\ & D & \end{tikzcd}$$

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 $e_{in} \circ m_{n+1} = f_{n+1i}$ and $\overline{e_n} \circ m_{n+1} = d_{n+1}$. Since (1_{n+1}) is a pushout and (5_{n+1}) is a pullback, by $\mathcal{M}$-pushout-pullback decomposition also (5_{n+1}) and (6_{n+1}) are pushouts, and it follows that $F_{n+1n+1} = E_n$. From pushout $(3_{in+1}$ 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 $q_{in+1} \circ h_{in+1} = h_{in} \circ p_{n+1}$ and $q_{in+1} \circ k_{in+1} = k_{in}$, and from pushout decomposition also (7_{in+1}) is a pushout.



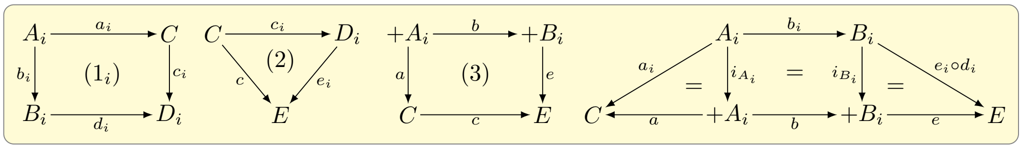
To show that (4_{n+1}) is a colimit, consider an object X and morphisms (x_i) and y with $x_i \circ h_{in+1} = y$ for $i = 1, \ldots, n$ and $x_{n+1} \circ p_{n+1} = y$. From pushout (7_{in+1}) we obtain a unique morphism z_i with $z_i \circ q_{in+1} = x_i$ and $z_i \circ h_{in} = x_{n+1}$. Now colimit (4_n) induces a unique morphism z with $z \circ \overline{e_n} = x_{n+1}$ and $z \circ l_{in} = z_i$. It 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 $z \circ \overline{e_{n+1}} = z \circ \overline{e_n} \circ p_{n+1} = x_{n+1} \circ p_{n+1} = y$. The uniqueness of z follows directly from the construction, thus (4_{n+1}) is the required colimit.



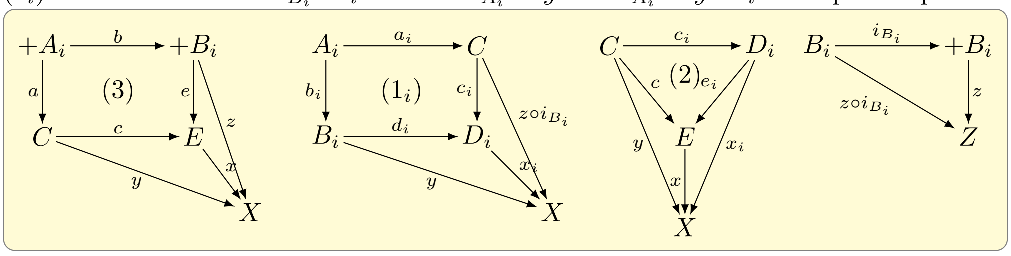
□

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

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



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



with $x_i \circ c_i = y$ and $x_i \circ d_i = z \circ i_{B_i}$. Now colimit (2) implies a unique morphism x with $x \circ c = y$ and $x \circ e_i = x_i$. It follows that $x \circ e \circ i_{B_i} = x \circ e_i \circ d_i = x_i \circ d_i = z \circ i_{B_i}$, and since z is unique w.r.t. $z \circ i_{B_i}$ it follows from the coproduct that $z = x \circ e$. Uniqueness of x follows from the uniqueness of x and x_i, and hence (3) is a pushout.

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

$$\begin{tikzcd}
A_i \arrow[r, "a_i"] \arrow[d, "b_i"'] \arrow[dr, phantom, "(1_i)"] & C \arrow[d, "c_i"] \arrow[ddr, "e\circ i_{B_i}"] & \\
B_i \arrow[r, "d_i"] \arrow[drr, "c"'] & D_i \arrow[dr, "e_i"] & \\
& & E
\end{tikzcd}
\quad
\begin{tikzcd}
C \arrow[rr, "c_i"] \arrow[dr, "c"] \arrow[ddr, "y"'] \arrow[rr, phantom, bend right, "(2)"] & & D_i \arrow[dl, "e_i"'] \arrow[ddl, "y_i"] \\
& E \arrow[d, "x"] & \\
& X &
\end{tikzcd}
\quad
\begin{tikzcd}
B_i \arrow[r, "i_{B_i}"] \arrow[dr, "y_i\circ d_i"'] & +B_i \arrow[d, "z"] \\
& X
\end{tikzcd}
\quad
\begin{tikzcd}
+A_i \arrow[r, "b"] \arrow[d, "a"'] \arrow[dr, phantom, "(3)"] & +B_i \arrow[d, "e"'] \arrow[ddr, "z"] & \\
C \arrow[r, "c"] \arrow[drr, "y"'] & E \arrow[dr, "x"] & \\
& & X
\end{tikzcd}$$

Given an object X and morphisms y, y_i with $y_i \circ c_i = y$ we obtain a morphism z with $z \circ i_{B_i} = y_i \circ d_i$ from coproduct $+B_i$. Then we have that $y \circ a \circ i_{A_i} = y_i \circ c_i \circ a_i = y_i \circ d_i \circ b_i = z \circ i_{B_i} \circ b_i = z \circ b \circ i_{A_i}$, and from coproduct $+A_i$ it follows that $y \circ a = z \circ b$. Now pushout (3) implies a unique morphism x with $x \circ c = y$ and $x \circ e = z$. From pushout (1_i) using $x \circ e_i \circ d_i = x \circ e \circ i_{B_i} = z \circ i_{B_i} = y_i \circ d_i$ and $x \circ e_i \circ c_i = x \circ c = y = y_i \circ c_i$ it follows that $x \circ e_i = y_i$, thus (2) is a colimit. □

Lemma B.7. Consider colimits (1) – (4) such that (5_i) is a pushout for all $i = 1, \ldots, n$ and (6_k) – (9_k) commute for all $k = 1, \ldots, m$. Then also (10) is a pushout.

$$\begin{tikzcd}
A_i \arrow[rr, "a_k"] \arrow[dr, "\overline{a_i}"'] & (1) & A_j \arrow[dl, "\overline{a_j}"] \\
& \overline{A} &
\end{tikzcd}
\quad
\begin{tikzcd}
B_i \arrow[rr, "b_k"] \arrow[dr, "\overline{b_i}"'] & (2) & B_j \arrow[dl, "\overline{b_j}"] \\
& \overline{B} &
\end{tikzcd}
\quad
\begin{tikzcd}
C_i \arrow[rr, "c_k"] \arrow[dr, "\overline{c_i}"'] & (3) & C_j \arrow[dl, "\overline{c_j}"] \\
& \overline{C} &
\end{tikzcd}
\quad
\begin{tikzcd}
D_i \arrow[rr, "d_k"] \arrow[dr, "\overline{d_i}"'] & (4) & D_j \arrow[dl, "\overline{d_j}"] \\
& \overline{D} &
\end{tikzcd}
\quad
\begin{tikzcd}
A_i \arrow[r, "f_i"] \arrow[d, "g_i"'] \arrow[dr, phantom, "(5_i)"] & B_i \arrow[d, "h_i"] \\
C_i \arrow[r, "k_i"'] & D_i
\end{tikzcd}$$

$$\begin{tikzcd}
A_i \arrow[r, "f_i"] \arrow[d, "a_k"'] \arrow[dr, phantom, "(6_k)"] & B_i \arrow[d, "b_k"] \\
A_j \arrow[r, "f_j"'] & B_j
\end{tikzcd}
\quad
\begin{tikzcd}
A_i \arrow[r, "g_i"] \arrow[d, "a_k"'] \arrow[dr, phantom, "(7_k)"] & C_i \arrow[d, "c_k"] \\
A_j \arrow[r, "g_j"'] & C_j
\end{tikzcd}
\quad
\begin{tikzcd}
B_i \arrow[r, "h_i"] \arrow[d, "b_k"'] \arrow[dr, phantom, "(8_k)"] & D_i \arrow[d, "d_k"] \\
B_j \arrow[r, "h_j"'] & D_j
\end{tikzcd}
\quad
\begin{tikzcd}
C_i \arrow[r, "k_i"] \arrow[d, "c_k"'] \arrow[dr, phantom, "(9_k)"] & D_i \arrow[d, "d_k"] \\
C_j \arrow[r, "k_j"'] & D_j
\end{tikzcd}
\quad
\begin{tikzcd}
\overline{A} \arrow[r, "\overline{f}"] \arrow[d, "\overline{g}"'] \arrow[dr, phantom, "(10)"] & \overline{B} \arrow[d, "\overline{h}"] \\
\overline{C} \arrow[r, "\overline{k}"'] & \overline{D}
\end{tikzcd}$$

Proof. The morphisms $\overline{f}$, $\overline{g}$, $\overline{h}$, and $\overline{k}$ are uniquely induced by the colimits. We show this exemplarily for the morphism $\overline{f}$. From colimit (1), with $\overline{b_j} \circ f_j \circ a_k = \overline{b_j} \circ b_k \circ f_i = \overline{b_i} \circ f_i$ we obtain a unique morphism $\overline{f}$ with $\overline{f} \circ \overline{a_i} = \overline{b_i} \circ f_i$. It follows directly that $\overline{k} \circ \overline{h} = \overline{h} \circ \overline{f}$.

$$\begin{tikzcd}
A_i \arrow[rr, "a_k"] \arrow[dr, "\overline{a_i}"'] \arrow[ddr, "\overline{b_i}\circ f_i"'] & (1) & A_j \arrow[dl, "\overline{a_j}"] \arrow[ddl, "\overline{b_j}\circ f_j"] \\
& \overline{A} \arrow[d, "\overline{f}"] & \\
& \overline{B} &
\end{tikzcd}
\quad
\begin{tikzcd}
\overline{A} \arrow[r, "\overline{f}"] \arrow[d, "\overline{g}"'] \arrow[dr, phantom, "(10)"] & \overline{B} \arrow[d, "\overline{h}"'] \arrow[ddr, "z"] & \\
\overline{C} \arrow[r, "\overline{k}"] \arrow[drr, "y"'] & \overline{D} \arrow[dr, "x"] & \\
& & X
\end{tikzcd}
\quad
\begin{tikzcd}
A_i \arrow[r, "f_i"] \arrow[d, "g_i"'] \arrow[dr, phantom, "(5_i)"] & B_i \arrow[d, "h_i"'] \arrow[ddr, "z\circ\overline{b_i}"] & \\
C_i \arrow[r, "k_i"] \arrow[drr, "y\circ\overline{c_i}"'] & D_i \arrow[dr, "x_i"] & \\
& & X
\end{tikzcd}
\quad
\begin{tikzcd}
D_i \arrow[rr, "d_k"] \arrow[dr, "\overline{d_i}"'] \arrow[ddr, "x_i"'] & (4) & D_j \arrow[dl, "\overline{d_j}"] \arrow[ddl, "x_j"] \\
& \overline{D} \arrow[d, "x"] & \\
& X &
\end{tikzcd}
\quad
\begin{tikzcd}
B_i \arrow[rr, "b_k"] \arrow[dr, "\overline{b_i}"'] \arrow[ddr, "z\circ\overline{b_i}"'] & (2) & B_j \arrow[dl, "\overline{b_j}"] \arrow[ddl, "z\circ\overline{b_j}"] \\
& \overline{B} \arrow[d, "z"] & \\
& X &
\end{tikzcd}$$

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

For all $k = 1, \ldots, m$, $x_j \circ d_k \circ k_i = x_j \circ k_j \circ c_k = y \circ \overline{c_j} \circ c_k = y \circ \overline{c_i}$ and $x_j \circ d_k \circ h_i = x_j \circ h_j \circ b_k = z \circ \overline{b_j} \circ b_k = z \circ \overline{b_i}$, and pushout (5_i) implies that $x_i = x_j \circ d_k$. This means that colimit (4) implies a unique x with $x \circ \overline{d_i} = x_i$. Now consider colimit (2), and $x \circ \overline{h} \circ \overline{b_i} = x \circ \overline{d_i} \circ h_i = x_i \circ h_i = z \circ \overline{b_i}$ implies that $x \circ \overline{h} = z$. Similarly, $x \circ \overline{k} = y$, and the uniqueness follows from the uniqueness of x with respect to (4). Thus, (10) is indeed a pushout. $\square$

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

$$
\begin{tikzcd}
A \arrow[rr, "a_i"] \arrow[dr, "\overline{a}"'] & (1) & A_i \arrow[dl, "\overline{a_i}"] \\
& \overline{A} &
\end{tikzcd}
\quad
\begin{tikzcd}
B \arrow[rr, "b_i"] \arrow[dr, "\overline{b}"'] & (2) & B_i \arrow[dl, "\overline{b_i}"] \\
& \overline{B} &
\end{tikzcd}
\quad
\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "a_i"'] \arrow[dr, phantom, "(3_i)"] & B \arrow[d, "b_i"] \\
A_i \arrow[r, "f_i"'] & B_i
\end{tikzcd}
\quad
\begin{tikzcd}
\overline{A} \arrow[r, "\overline{f}"] \arrow[d, "\overline{c}"'] \arrow[dr, phantom, "(4)"] & \overline{B} \arrow[d, "\overline{d}"] \\
C \arrow[r, "e"'] & D
\end{tikzcd}
\quad
\begin{tikzcd}
+A_i \arrow[r, "+f_i"] \arrow[d, "c"'] \arrow[dr, phantom, "(5)"] & +B_i \arrow[d, "d"] \\
C \arrow[r, "e"'] & D
\end{tikzcd}
$$

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

$$
\begin{tikzcd}
A \arrow[rr, "a_i"] \arrow[dr, "\overline{a}"'] \arrow[ddr, "\overline{b}\circ f"'] & (1) & A_i \arrow[dl, "\overline{a_i}"] \arrow[ddl, "\overline{b_i}\circ f_i"] \\
& \overline{A} \arrow[d, "\overline{f}"] & \\
& \overline{B} &
\end{tikzcd}
$$

We have to show that (5) is a pushout. Given morphisms x, y with $x \circ c = y \circ (+f_i)$, we have that $y \circ i_{B_i} \circ b_i \circ f = y \circ i_{B_i} \circ f_i \circ a_i = y \circ (+f_i) \circ i_{A_i} \circ a_i = x \circ c \circ i_{A_i} \circ a_i = x \circ \overline{c} \circ \overline{a_i} \circ a_i = x \circ \overline{c} \circ \overline{a}$ for all $i = 1, \ldots, n$. f being an epimorphisms implies that $y \circ i_{B_i} \circ b_i = y \circ i_{B_j} \circ b_j$ for all i, j. Now define $y' := y \circ i_{B_i} \circ b_i$, and from colimit (2) we obtain a unique morphism $\overline{y}$ with $\overline{y} \circ \overline{b_i} = y \circ i_{B_i}$ and $\overline{y} \circ \overline{b} = y'$.

$$
\begin{tikzcd}
A_i \arrow[r, "i_{A_i}"] \arrow[dr, "\overline{c}\circ\overline{a_i}"'] & +A_i \arrow[d, "c"] \\
& C
\end{tikzcd}
\quad
\begin{tikzcd}
B_i \arrow[r, "i_{B_i}"] \arrow[dr, "\overline{d}\circ\overline{b_i}"'] & +B_i \arrow[d, "d"] \\
& D
\end{tikzcd}
\quad
\begin{tikzcd}
A_i \arrow[r, "i_{A_i}"] \arrow[d, "f_i"'] & +A_i \arrow[d, "+f_i"] \\
B_i \arrow[r, "i_{B_i}"'] & +B_i
\end{tikzcd}
$$

$$
\begin{tikzcd}
+A_i \arrow[r, "+f_i"] \arrow[d, "c"'] \arrow[dr, phantom, "(5)"] & +B_i \arrow[d, "d"] \arrow[ddr, "y"] & \\
C \arrow[r, "e"] \arrow[drr, "x"'] & D \arrow[dr, "z"] & \\
& & X
\end{tikzcd}
\quad
\begin{tikzcd}
B \arrow[rr, "b_i"] \arrow[dr, "\overline{b}"'] \arrow[ddr, "y'"'] & (2) & B_i \arrow[dl, "\overline{b_i}"] \arrow[ddl, "y\circ i_{B_i}"] \\
& \overline{B} \arrow[d, "\overline{y}"] & \\
& X &
\end{tikzcd}
\quad
\begin{tikzcd}
A \arrow[rr, "a_i"] \arrow[dr, "\overline{a}"'] \arrow[ddr, "\overline{y}\circ\overline{f}\circ\overline{a}"'] & (1) & A_i \arrow[dl, "\overline{a_i}"] \arrow[ddl, "\overline{y}\circ\overline{f}\circ\overline{a_i}"] \\
& \overline{A} \arrow[d, "\overline{y}\circ\overline{f}"] & \\
& X &
\end{tikzcd}
\quad
\begin{tikzcd}
\overline{A} \arrow[r, "\overline{f}"] \arrow[d, "\overline{c}"'] \arrow[dr, phantom, "(4)"] & \overline{B} \arrow[d, "\overline{d}"] \arrow[ddr, "\overline{y}"] & \\
C \arrow[r, "e"] \arrow[drr, "x"'] & D \arrow[dr, "z"] & \\
& & X
\end{tikzcd}
$$

Now $x \circ \overline{c} \circ \overline{a_i} = x \circ c \circ i_{A_i} = y \circ (+f_i) \circ i_{A_i} = y \circ i_{B_i} \circ f_i = \overline{y} \circ \overline{b_i} \circ f_i = \overline{y} \circ \overline{f} \circ \overline{a_i}$ and $x \circ \overline{c} \circ \overline{a} = x \circ \overline{c} \circ \overline{a_i} \circ a_i = \overline{y} \circ \overline{f} \circ \overline{i} \circ a_i = \overline{y} \circ \overline{f} \circ \overline{a}$, and the uniqueness of the induced colimit morphism implies that $\overline{y} \circ \overline{f} = x \circ \overline{c}$. This means that X can be compared to pushout (4), and we obtain a unique morphism z with $z \circ \overline{d} = \overline{y}$ and $z \circ e = x$. Now $z \circ d \circ i_{B_i} = z \circ \overline{d} \circ \overline{b_i} = \overline{y} \circ \overline{b_i} = y \circ i_{B_i}$, and it follows that $z \circ d = y$. Similarly, the uniqueness of z w.r.t. the pushout propert of (5) follows, thus (5) is a pushout. $\square$