Chapter 6
Multi-amalgamated Transformations

In this chapter, we introduce amalgamated transformations. An amalgamated rule is based on a kernel rule, which defines a fixed part of the match, and multi rules, which extend this fixed match. From a kernel and a multi rule, a complement rule can be constructed which characterises the effect of the multi rule exceeding the kernel rule. If multiple rules can be applied using the same kernel rule, as a first main result the Multi-amalgamation Theorem states that a bundle of $s$-amalgamable transformations is equivalent to a corresponding amalgamated transformation. An interaction scheme is defined by a kernel rule and available multi rules, leading to a bundle of multi rules that specifies in addition how often each multi rule is applied. Amalgamated rules are in general standard rules in $\mathcal{M}$-adhesive transformation systems; thus all the results follow. In addition, we are able to refine parallel independence of amalgamated rules based on the induced multi rules. If we extend an interaction scheme as large as possible we can describe the transformation for an unknown number of matches, which otherwise would have to be defined by an infinite number of rules. This leads to maximal matchings, which are useful for defining the semantics of models. For this chapter, we require an $\mathcal{M}$-adhesive category with binary coproducts as well as initial and effective pushouts (see Sect. 4.3). The theoretical results in this chapter are based on [GHE14].

In Sect. 6.1, kernel, multi and complement rules are presented. In Sect. 6.2, we introduce amalgamated rules and transformations and show some important results in Sect. 6.3. In Sect. 6.4, we define interaction schemes and maximal matching and use these concepts for the firing semantics of elementary Petri nets modelled by typed graphs using amalgamation. This chapter is based on [EGH+14, Gol11].

6.1 Kernel Rules, Multi Rules, and Complement Rules

In the following, a bundle represents a family of morphisms or transformation steps with the same domain, which means that a bundle of things always starts at the same object.
A kernel morphism describes how a smaller rule, the kernel rule, is embedded into a larger rule, the multi rule. The multi rule has its name because it can be applied multiple times for a given kernel rule match, as described later. 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 us to construct a complement rule.

**Definition 6.1 (Kernel morphism).** Given rules $p_0 = (L_0 \xrightarrow{l_0} K_0 \xrightarrow{r_0} R_0, a_{c_0})$ and $p_1 = (L_1 \xrightarrow{l_1} K_1 \xrightarrow{r_1} R_1, a_{c_1})$, a kernel morphism $s_1 : p_0 \rightarrow p_1$, $s_1 = (s_{1,L}, s_{1,K}, s_{1,R})$ consists of $M$-morphisms $s_{1,L} : L_0 \rightarrow L_1$, $s_{1,K} : K_0 \rightarrow K_1$, and $s_{1,R} : R_0 \rightarrow R_1$ such that the diagrams $(1)$ and $(2)$ are pullbacks, $(1)$ has a pushout complement $(1')$ for $s_{1,L} \circ l_0$, and $a_{c_1} \Rightarrow \text{Shift}(s_{1,L}, a_{c_0})$. In this case, $p_0$ is called kernel rule and $p_1$ multi rule.

$ac_0$ and $ac_1$ are complement-compatible w. r. t. $s_1$ if there is some application condition $ac_1'$ on the pushout complement $L_{10}$ such that $ac_1 \cong \text{Shift}(s_{1,L}, ac_0) \land L(p_1^*, \text{Shift}(v_1, ac_1'))$ for the pushout $(3)$ and $p_1' = (L_1 \xleftarrow{l_1} L_{10} \xrightarrow{v_1} E_1)$.

**Remark 6.2.** The complement-compatibility makes sure that there is a decomposition of $ac_1$ into parts on $L_0$ and $L_{10}$. The latter are used later for the application conditions of the complement rule, which ensure the equivalence of the composition.

**Example 6.3.** 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 have different labels for edges—black and dotted ones. The kernel rule $p_0$ is depicted in Fig. 6.1. It selects a node with a black loop, deletes this loop, and adds a dotted loop, all of this if no dotted loop is already present. The matches are defined by the numbers at the nodes and can be induced for the edges by their position.

In Fig. 6.2, two multi rules $p_1$ and $p_2$ are shown which extend the rule $p_0$ and in addition reverse an edge if no backward edge is present. They also inherit the application condition of $p_0$, forbidding a dotted loop at the selected node. There is a kernel morphism $s_1 : p_0 \rightarrow p_1$, as shown in the top of Fig. 6.2, with pullbacks $(1)$, $(2)$ and pushout complement $(1')$. Similarly, there is a kernel morphism $s_2 : p_0 \rightarrow p_2$, as shown in the bottom of Fig. 6.2, with pullbacks $(1_2)$, $(2_2)$ and pushout complement $(1'_2)$.

For the application conditions, it holds that $ac_1 = \text{Shift}(s_{1,L}, ac_0) \land \neg \exists a_1 \cong \text{Shift}(s_{1,L}, ac_0) \land L(p_1^*, \text{Shift}(v_1, \neg \exists a_1'))$ with $a_1'$ as shown in the left of Fig. 6.3. We have that $\text{Shift}(v_1, \neg \exists a_1') = \neg \exists a_{11}$, because square $(*)$ is the only possible commuting square leading to morphism $(a_{11}, b_{11})$ being jointly surjective and $b_{11}$