Chapter 7
Model Transformation and Model Integration

In this chapter, we describe the formal framework for model transformation and model integration based on triple graph grammars. For this purpose, we use triple graph transformation systems as introduced in Chap. 3 and show in Sect. 7.1 that they instantiate the general framework of \( \mathcal{M} \)-adhesive transformation systems presented in Chap. 5. This ensures that all results for \( \mathcal{M} \)-adhesive transformation systems hold for the specific case of triple graph transformation systems. A triple graph grammar is a constructive specification of a language of integrated models, which are specified by their underlying abstract syntax graphs. Based on this general concept, we first derive a transformation system for forward model transformations, which are defined in Sect. 7.3. In Sect. 7.4, we introduce forward translation rules as an alternative to forward rules and show the equivalence of model transformations based on either forward or forward translation rules. The concept of forward translation rules simplifies the control mechanism for executing model transformations. In addition to that, it offers improved capabilities for analysis and execution, which we will study in detail in Chap. 8. Model integration is a technique to integrate two given models—one from the source and one from the target language. In Sect. 7.5, we present model integration based on TGGs and show formally that this concept is closely related to model transformations. The last section (Sect. 7.6) of this chapter relates the presented concepts based on TGGs with standard model transformations based on plain graph grammars. The chapter is based on [Her11, HEGO14, EEE+07, EEH08c, HHK10, GEH11].

7.1 Triple Graphs form an \( \mathcal{M} \)-adhesive Category

A triple graph is an integrated model \( G = (G^S \leftarrow G^C \rightarrow G^T) \) containing a source model \( G^S \) from the source language, a target model \( G^T \) from the target language, and explicit correspondences between them specified via a correspondence model \( G^C \).
The category of triple graphs can be constructed from underlying $M$-adhesive categories of typed attributed graphs.

**Fact 7.1 (Construction of categories of triple graphs).** The category of typed attributed triple graphs in Def. 3.4 and the base categories for triple graphs in Def. 3.5 can be constructed as follows.

- The category $\text{TrGraphs}$ of triple graphs and triple graph morphisms can be constructed as functor category $[X, \text{Graphs}]$ over the category $\text{Graphs}$ of graphs with schema category $X = ((S, C, T), \{s: C \to S, t: C \to T, \text{id}_S, \text{id}_C, \text{id}_T\})$.

- The category $\text{ATrGraphs}$ of attributed triple graphs can be constructed as the functor category $[X, \text{AGraphs}]$ over the category $\text{AGraphs}$ of attributed graphs with the same schema category as above.

- The category $\text{TrGraphs}_{TG}$ of typed triple graphs (or $\text{ATrGraphs}_{ATG}$ of typed attributed triple graphs) for a given triple graph $TG$ in $\text{TrGraphs}$ (or $ATG$ in $\text{ATrGraphs}$) can be constructed as slice category $\text{TrGraphs}_{nTG}$ over $\text{TrGraphs}$ (or $\text{ATrGraphs}_{nATG}$ over $\text{ATrGraphs}$).

**Proof.** A triple graph $G = (G^S \xleftarrow{s_G} G^C \xrightarrow{t_G} G^T)$ is represented by a functor $G: X \to \text{Graphs}$ with $G(S) = G^S$, $G(C) = G^C$, $G(T) = G^T$ and $G(s) = s_G$, $G(t) = t_G$. The compatibility condition for triple graphmorphisms follows from the compatibility condition of functor transformations that form the morphisms in a functor category. The typing and attribution extensions are compatible with the construction of the functor category.

**Theorem 7.2 (Category of triple graphs is $M$-adhesive).** The categories $\text{TrGraphs}$, $\text{TrGraphs}_{TG}$, $\text{ATrGraphs}$, and $\text{ATrGraphs}_{ATG}$ are $M$-adhesive.

**Proof.** Using Fact 7.1, we derive the categories as functor and slice constructions over $M$-adhesive categories ($\text{Graphs}$, $M$) and ($\text{AGraphs}$, $M$) and can apply Theorem B.13 to derive that the constructed categories are again $M$-adhesive categories.

By Theorem 7.2, we can conclude that the results in Chapters 4 to 6 for $M$-adhesive categories hold for triple graphs and triple graph transformations. In particular, we will apply the theory and analysis for critical pairs and confluence (see Sect. 5.2.4 in Chap. 5) in Chap. 8 for analysing functional behaviour and information preservation. Using Theorem 7.2, we derive the classes of $M$-morphisms for the different kinds of triple graphs as constructions from the $M$-adhesive categories $\text{Graphs}$ and $\text{AGraphs}$. The class of $M$-morphisms is given by all triple graph morphisms that are injective on the graph part and isomorphisms on the data part.

From the application point of view a model transformation should be injective on the structural part, i.e., the transformation rules are applied along matches that do not identify structural elements. Thus, the translation of each element is explicitly specified and there is no confusion. But it would be too restrictive to require injectivity of the matches also on the data and variable nodes, because we must allow two different variables to be mapped to the same data value. For this reason we introduce the notion of almost injective matches, which requires matches