A Component Concept for System Modeling Based on High-Level Replacement Transformations

Markus Klein
email: klein@cs.tu-berlin.de

April 2003

Abstract

The present work is a theoretical foundation of the application of an abstract transformation based component concept to high-level replacement systems, that can be understood as a categorical abstraction of graph transformation systems. This is done by proving the assumptions of the abstract framework concerning the underlying transformations for the case of high-level replacement transformations. These assumptions describe the existence of an extension for given transformations and morphisms and based on that the compositionality and decomposition of extension diagrams. The corresponding proofs assume several properties for the different categories. The result of the application is a component concept that can be used to specify software systems with categorical techniques in a structured way. This concept is then applied to three sample categories what means to show that the assumptions of the proofs for the properties of high-level replacement transformations are valid in the concrete categories. The first example deals with classical place/transition nets while the other two deal with algebraic specifications and algebraic high-level nets, respectively. In all cases a sample system with components of the corresponding category is specified. In the case of the place/transition nets and the algebraic high-level nets the example's domain is the automation of production. The example for the algebraic specifications models a total automaton.

Contents

Chapter 1

Introduction

Component Based software design and implementing approaches become more and more important in today's applied software engineering. Technologies as Corba or Java Beans found a wide acceptance in the software developers community. Especially the possibility of integrating program parts written in different programming languages or running them on different machines seems to be very powerful. Since one main feature of component based architectures is the possibility to develop components independently from their environment a component approach should offer the possibility to formulate abstract interfaces. On that way it would also be possible to perform *encapsulated* system changes in the sense of [16]. The feature of developing components independently from each other would also be eased by a formal semantics for single components. Of course, such a semantic should be somehow consistent with respect to the composition of components. On this way it would be possible to detect errors in the specification of the component based system in the pre-implementation phase already.

Unfortunately, the known component based specification techniques as for example [2] lack such abstract interfaces and formal semantics. The interfaces are always connected to the environment of a component. In contrast to that the abstract component framework in [12, 13] provides the features mentioned above. This report now covers one possible instantiation of the abstract framework. More specific, it provides the theoretical background for the conceptual instantiation to high-level replacement (HLR) Systems presented in [11], the result is a transformation based component concept for HLR categories.

An interesting property of this component framework is its inner structure. In contrast to other structuring techniques with formal semantics and abstract interfaces, as for example [9], there is no constructive relation between the import and the body. The idea is to *refine* the export interface along a transformation into the body. The import interface is then just included in the body. This approach comes quite near to a top down development strategy while the constructor based approaches seem to be more adequate for bottom up strategies. Moreover, the refinement based approach seems to be more adequate for techniques that focus on the behavior of a system and the constructor based approach seems to be more adequate for the specification of data types.

Additionally, the report covers three instantiations of high-level replacement systems as instantiations of the component concept. The first one is the category of place/transition nets, the second one is a form of algebraic specifications while the third one is the category of algebraic high-level nets.

The instantiation of the abstract framework to HLR transformations seemed senseful because of the existing theoretical foundations for HLR categories. In this context it is important to mention the three famous theorems on parallel transformations: local church-rosser theorem one and two and the parallelism theorem. Moreover, we have the concurrency theorem that covers transformations that are not sequentially independent and we have different kinds of embedding theorems. All these theorems use different sets of assumptions concerning the underlying categories. These different HLR conditions reach from assuming different forms of limits and colimits to complicated pushout pullback decomposition properties. In[19] you can find an overview of the theorems including their assumptions and the categories that fulfill the different conditions.

These results are then applicable to the objects in the components. Moreover, there is a large variety of techniques that can be expressed in terms of HLR systems. Algebraic high level nets, algebraic specifications and statecharts are only some of them. In all cases it would be sufficient to show that the different conditions for the theorems hold to obtain a component concept corresponding to this report or to have the validity of the theorems mentioned above.

The report is organized as follows. Chapter 2 introduces the abstract component concept in the sense of [12]. Chapter 3 first of all repeats the main notions for HLR systems and the results needed for the rest of the report. After that some additional properties needed for the instantiation of the abstract component concept are formulated and proven. Additionally section 3.5 presents the resulting component concept for HLR categories. Chapters 4, 5 and 6 present three possible instantiations of the concept of the previous chapter. In the first two cases this is done by proving several properties that were assumed in chapter 3, while Chapter 6 only sketches the formal instantiation. Each instantiation example finishes with some sample components. The last chapter summarizes the main efforts of the report and gives an outlook on work that could be done to extent the results.

Acknowledgements: This work has been supported by the Deutsche Forschungsgemeinschaft (German Research Council), project IOSIP within the DFG Priority Programme "Integration of Software Specification Techniques for Applications in Engineering". Thanks to Hartmut Ehrig, Fernando Orejas, Benjamin Braatz and Martti Piirainen for fruitful discussions and hints concerning the report.

Chapter 2

The Abstract Transformation Based Component Framework

This chapter gives a description of the abstract component framework this work is mainly based on. This concept was initiated by Hartmut Ehrig and Fernando Orejas and formulated in [12, 13].

In this context abstract means that the concept is generic with respect to the used specification technique and with respect to the underlying notion of transforming specification parts. A detailed discussion of these aspects is given in sect. 2.1 and in sect. 2.2 respectively. In order to construct larger systems with components the concept has to handle the connection of components. This composition of components is explained in sect. 2.4. The concept gives semantics to the components in terms of transformation mappings which turn out to be compositional. This topic is covered by sect. 2.3.

2.1 The Generic Component Concept

The main idea of this component approach is to give a notion to construct a larger system from smaller units, the components. In order to ease this construction of a larger system it is important to distinguish between the interface part and the body part of a component. The interface part describes what functionality is offered to or needed from the components environment. Following these ideas a *component specification* is defined like follows.

Definition 2.1.1 (Component) *A* component specification (short component) *$COMP$ is defined as $COMP = (IMP, EXP, BOD, imp, exp)$, where:*

- *IMP is called* import interface*,*
- *EXP is called* export interface*,*
- *BOD is called* body*,*
- *exp defines a transformation from EXP to BOD and*
- *imp defines an inclusion from IMP to BOD.*

$$
\begin{tikzcd}
 & EXP \arrow[d, Rightarrow, "exp"] \\
IMP \arrow[r, "imp"] & BOD
\end{tikzcd}
$$

It should be mentioned that this approach is different from classical constructor based component and module approaches as for example the algebraic module specification in the sense of [9]. All these approaches follow the aim of constructing the body and the export parts of a component from an instantiation of the import part. Here the general idea is that the export part of a component is refined in the body part which includes an import part that can be further refined.

2.2 The Generic Transformation Concept

In the last section a general description of the components in this framework was given. One important part of a component is the transformation from the export to the body part of the component. Moreover, transformations will be used to connect components. This topic is further explained in sect. 2.4. The concept leaves open, besides some general assumptions, what these transformations have to look like. The assumptions on the general transformation framework are:

- it consists of a class of transformations,
- it includes identical transformations,
- it is closed under composition,

- it satisfies the *extension property* defined below,
- it satisfies the *horizontal and vertical compositionality of extension diagrams* as defined below and
- it satisfies the *horizontal and vertical decomposition of extension diagrams* as defined below.

Definition 2.2.1 (Extension Property) *A transformation framework satisfies the* extension property*, if for each transformation $t : A \Rightarrow B$ and for each inclusion $i_1 : A \to A'$, where i_1 has to be consistent with respect to t, there is a selected transformation $t' : A' \Rightarrow B'$ and a selected inclusion $i_2 : B \to B'$. This situation leads to an* extension diagram *as depicted below.*

$$\begin{tikzcd} A \arrow[r, Rightarrow, "t"] \arrow[d, "i_1"] & B \arrow[d, "i_2"] \\ A' \arrow[r, Rightarrow, "t'"] & B' \end{tikzcd}$$

We call t' together with i_2 extension *of t and i_1.*

Remark: Note that this formulation of the the extension property is more general than the one used in [12] and [13]. In that papers the constraints on the existence of an extension for a transformation and an inclusion are mentioned but they are not used in the definition of the extension property. But we have to use a consistency condition here because in the case of HLR systems we do not have such a strong extension property. At this point the abstract framework leaves open how this consistency is defined. For HLR systems the consistency corresponds to the fact that a transformation preserves the boundary of a morphism (sect. 3.1). But in the general case it is also possible to define that all pairs of morphisms and inclusions fulfill the consistency condition, what would correspond to the strong extension property.

Definition 2.2.2 (Horizontal Compositionality) *A transformation framework satisfies the* horizontal compositionality of extension diagrams *if the existence of the extension diagrams (1) and (2) below implies that the diagram (1)+(2) is an extension diagram as well.*

$$\begin{tikzcd} G_0 \arrow[r, Rightarrow, "t_1"] \arrow[d, "d_0"] \arrow[dr, phantom, "(1)"] & G_n \arrow[r, Rightarrow, "t_2"] \arrow[d, "d_n"] \arrow[dr, phantom, "(2)"] & G_m \arrow[d, "d_m"] \\ G'_0 \arrow[r, Rightarrow, "t'_1"] & G'_n \arrow[r, Rightarrow, "t'_2"] & G'_m \end{tikzcd}$$

Definition 2.2.3 (Vertical Compositionality) *A transformation framework satisfies the* vertical compositionality of extension diagrams *if the existence of the extension diagrams (1) and (2) below implies that the diagram (1)+(2) is an extension diagram as well.*

$$
\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t"] \arrow[d, "d_0"'] \arrow[dr, phantom, "(1)"] & G_z \arrow[d, "d_z"] \\
G'_0 \arrow[r, Rightarrow, "t'"] \arrow[d, "d'_0"'] \arrow[dr, phantom, "(2)"] & G'_z \arrow[d, "d'_z"] \\
G''_0 \arrow[r, Rightarrow, "t''"] & G''_z
\end{tikzcd}
$$

Definition 2.2.4 (Horizontal Decomposition) *A transformation framework provides the* horizontal decomposition of extension diagrams *if the extension diagram (1) below can be decomposed into the extension diagrams (2) and (3).*

$$
\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t_1"] \arrow[d, "d_0"'] & G_n \arrow[r, Rightarrow, "t_2"] \arrow[d, phantom, "(1)"] & G_m \arrow[d, "d_m"] \\
G'_0 \arrow[rr, Rightarrow, "t'"] & & G'_m
\end{tikzcd}
\qquad
\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t_1"] \arrow[d, "d_0"'] \arrow[dr, phantom, "(2)"] & G_n \arrow[r, Rightarrow, "t_2"] \arrow[d, "d_n"] \arrow[dr, phantom, "(3)"] & G_m \arrow[d, "d_m"] \\
G'_0 \arrow[r, Rightarrow, "t'_1"] & G'_n \arrow[r, Rightarrow, "t'_2"] & G'_m
\end{tikzcd}
$$

Definition 2.2.5 (Vertical Decomposition) *A transformation framework provides the* vertical decomposition of extension diagrams *if the extension diagram (1) below can be decomposed into the extension diagrams (2) and (3).*

$$
\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t"] \arrow[d, "d_0"'] & G_n \arrow[dd, "k_n"] \\
G'_0 \arrow[d, "d'_0"'] \arrow[r, phantom, "(1)"] & \\
G''_0 \arrow[r, Rightarrow, "t''"] & G''_n
\end{tikzcd}
\qquad
\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t"] \arrow[d, "d_0"'] \arrow[dr, phantom, "(2)"] & G_n \arrow[d, "d_n"] \\
G'_0 \arrow[r, Rightarrow, "t'"] \arrow[d, "d'_0"'] \arrow[dr, phantom, "(3)"] & G'_n \arrow[d, "d'_n"] \\
G''_0 \arrow[r, Rightarrow, "t''"] & G''_n
\end{tikzcd}
$$

2.3 Transformation Semantics

Since this transformation based approach does not support a model semantics as for example many constructor based approaches a new kind of semantics has to be defined. In [13] the general idea is described as giving semantics to components by combining each possible import transformation $tra : IMP \Rightarrow SPEC$ with the export transformation $exp : EXP \Rightarrow BOD$. This combination is achieved by the application of the extension property. In each component we have an inclusion $imp : IMP \rightarrow BOD$. Now we consider an arbitrary transformation $tra : IMP \Rightarrow SPEC$, such that imp is consistent with respect to tra. We obtain a selected transformation $tra' : BOD \Rightarrow SPEC'$ with an inclusion $imp' : SPEC \rightarrow SPEC'$.

$$\begin{tikzcd}
& EXP \arrow[d, Rightarrow, "exp"] \\
IMP \arrow[r, "imp"] \arrow[d, Rightarrow, "tra"] \arrow[dr, phantom, "(*)"] & BOD \arrow[d, Rightarrow, "tra'"] \\
SPEC \arrow[r, "imp'"] & SPEC'
\end{tikzcd}$$

Now $Trafo(IMP)$ denotes the class of all possible transformations $tra : IMP \Rightarrow SPEC$ and $Trafo(Exp)$ denotes the class of all possible transformations $tra : EXP \Rightarrow SPEC$.

Definition 2.3.1 (Transformation Semantics) *The* transformation semantics *of a component $COMP$ is a partial function*

$$TrafoSem(COMP) : Trafo(IMP) \nrightarrow Trafo(EXP)$$

defined for all transformations tra such that imp is consistent with respect to tra what leads to the extension diagram (*) above. Then $TrafoSem(COMP)$ is defined by the following equation:

$$TrafoSem(COMP)(tra) = tra' \circ exp.$$

2.4 Composition of Components

Since the general idea of a component concept is to create a larger system from the components it is necessary to have a possibility to connect components. Moreover, it would be desirable if this connection would lead to a composition of components. That means if we connect two components we obtain a new, larger component. Fortunately, this specific concept supports this kind of connections. Although there are concepts for other connection operations already the main focus here lies on the hierarchical composition.

Since the extension property was assumed to be valid for transformations and consistent morphisms we need one additional property for the composition.

Definition 2.4.1 (Composability) *Given two components $COMP_1$ and $COMP_2$ with $COMP_i = (IMP_i, EXP_i, imp_i, exp_i)$ for $i = 1, 2$ and a transformation connect : $IMP_1 \Rightarrow EXP_2$. We call $COMP_1$ and $COMP_2$* composable via *connect if $imp_1 : IMP_1 \to BOD_1$ is consistent with respect to the transformation $connect; exp_2$.*

Definition 2.4.2 (Composition) *Given components $COMP_1$, $COMP_2$ and a transformation connect : $IMP_1 \Rightarrow EXP_2$ such that $COMP_1$ is composable with $COMP_2$ via connect, where*

$$COMP_i = (IMP_i, EXP_i, BOD_i, imp_i, exp_i)$$

for $i \in \{1, 2\}$. Let $xconnect = exp_2 \circ connect$. Since imp_1 is consistent with respect to xconnect, we have $xconnect' : BOD_1 \Rightarrow BOD_3$, $imp'_1 : BOD_2 \to BOD_3$ as extension of xconnect. Then we define the composition $COMP_3 = COMP_1 \circ_{connect} COMP_3$ as

$$COMP_3 = (IMP_3, EXP_3, BOD_3, imp_3, exp_3),$$

where $imp_3 = imp'_1 \circ imp_2$ and $exp_3 = xconnect' \circ exp_1$.

The diagram below shows how a component $COMP_3$ is constructed as composition of components $COMP_1$ and $COMP_2$ via *connect*.

$$\begin{tikzcd}
 & & EXP_3 = EXP_1 \arrow[d, Rightarrow, "exp_1"] \\
 & IMP_1 \arrow[r, "imp_1"] \arrow[d, Rightarrow, "connect"] & BOD_1 \arrow[dd, Rightarrow, "xconnect'"] \\
 & EXP_2 \arrow[d, Rightarrow, "exp_2"] & \\
IMP_3 = IMP_2 \arrow[r, "imp_2"] & BOD_2 \arrow[r, "imp'_1"] & BOD_3
\end{tikzcd}$$

2.5 Compositionality of Transformation Semantics

This section sums up one important result of the component concept. Compositionality of transformation semantics means that the result is the same if we:

- compute the semantics of two connected components separately and then compose these two functions, or
- compose the new component via the connection and then take the semantics of this new component.

Theorem 2.5.1 (Compositionality of Transformation Semantics) *Given a transformation framework with horizontal and vertical compositionality and decomposition of extension diagrams and given a composition of components* $COMP_3 = COMP_1 \circ_{connect} COMP_2$ *then we have*

$$\begin{aligned}&TrafoSem(COMP_3) = \\ &TrafoSem(COMP_1) \circ Trafo(connect) \circ TrafoSem(COMP_2),\end{aligned}$$

where $Trafo(connect)$ *is defined as* $Trafo(connect)(tra) = tra \circ connect$.

Remark: Since the two functions above are partial the stated equality means that both functions are defined for the same arguments and they deliver the same result if they are defined.

Proof: Given a composition of components $COMP_3 = COMP_1 \circ_{connect} COMP_2$ with $COMP_i = (IMP_i, EXP_i, BOD, imp_i, exp_i)$ for $i = 1,2,3$ and a transformation $trafo_3 : IMP_3 \Rightarrow SPEC_3$ as depicted below, where diagram (1) is an extension diagram by the definition of $COMP_3$.

$$\begin{tikzcd}
 & & EXP_3 = EXP_1 \arrow[d, Rightarrow, "exp_1"] \\
 & IMP_1 \arrow[r, "imp_1"] \arrow[d, Rightarrow, "connect"] & BOD_1 \arrow[dd, Rightarrow, "xconnect'"] \\
 & EXP_2 \arrow[r, phantom, "(1)"] \arrow[d, Rightarrow, "exp_2"] & \\
IMP_3 = IMP_2 \arrow[r, "imp_2"] \arrow[d, Rightarrow, "trafo_3"] \arrow[dr, phantom, "(3)"] & BOD_2 \arrow[r, "imp'_1"] \arrow[d, Rightarrow, "trafo_2"] \arrow[dr, phantom, "(2)"] & BOD_3 \arrow[d, Rightarrow, "trafo_1"] \\
SPEC_3 \arrow[r, hook] & SPEC_2 \arrow[r, hook] & SPEC_1
\end{tikzcd}$$

First, it is necessary to show that the two functions are defined for the same set of transformations. This is done by showing that if one of the two functions is defined the other one is defined as well.

1. Let $TrafoSem(COMP_3)(trafo_3)$ be defined. By the definition of this function we can conclude that (3)+(2) is an extension diagram. If we now apply the vertical decomposition we obtain that (2) and (3) are extension diagrams. By the definition of $COMP_3$ we know that (1) is an extension diagram. Now we can compose the extension diagram (1)+(2) and so we know that $TrafoSem(COMP_1) \circ Trafo(connect) \circ TrafoSem(COMP_2)$ is defined for $(trafo_3)$.

2. Now we assume that $TrafoSem(COMP_1) \circ Trafo(connect) \circ TrafoSem(COMP_2)$ is defined for $(trafo_3)$. This means that we have the extension diagrams (3) and (1)+(2). Using the vertical decomposition we obtain that (2) is an extension diagram. So we can compose the extension diagram (2)+(3). Since we know that (1) is an extension diagram we know that $TrafoSem(COMP_3)(trafo_3)$ is defined.

Now it remains to show that both functions give the same results if they are defined. Diagram (3) is then constructed as extension of $trafo_3$ with respect to imp_2 and Diagram (2) is constructed as extension of $trafo_2$ with respect to imp'_1. Using the compositionality of extension diagrams horizontally and vertically implies that (1+2) and (2+3) are extension diagrams as well. Since we have $imp_3 = imp'_1 \circ imp_2$ and $exp_3 = xconnect' \circ exp_1$ and the extension diagram (2+3) we know that

$$TrafoSem(COMP_3)(trafo_3) = trafo_1 \circ xconnect' \circ exp_1.$$

For the other side of the theorem's equation we have:

$$\begin{aligned}
& TrafoSem(COMP_1) \circ Trafo(connect) \circ \\
& TrafoSem(COMP_2)(trafo_3) \\
= \; & TrafoSem(COMP_1) \circ Trafo(connect)(trafo_2 \circ exp_2) \\
& \text{(by definition of } TrafoSem(COMP_2)) \\
= \; & TrafoSem(COMP_1)(trafo_2 \circ exp_2 \circ connect) \\
& \text{(by definition of } Trafo(connect)) \\
= \; & trafo_1 \circ xconnext' \circ exp_1 \\
& \text{(by extension diagram (1+2))}.
\end{aligned}$$

This implies that

$$\begin{aligned}
& Trafosem(COMP_3) = \\
& TrafoSem(COMP_1) \circ Trafo(connect) \circ TrafoSem(COMP_2). \quad \square
\end{aligned}$$

2.6 Associativity of Composition

This section covers another result concerning the composition of components. It states that a hierarchical composition of three components is the same if we:

- compute the composition for the two upper components and then compose the result with the lower component or
- compute the composition for the two lower components and then compose the upper component with the result of the first step.

Theorem 2.6.1 (Associativity of Composition) *Given the components*

$$COMP_i = (IMP_i, EXP_i, BOD, imp_i, exp_i)$$

for $i = 1, 2, 3$ *and transformations* $trafo_i : IMP_i \Rightarrow EXP_{i+1}$ *for* $i = 1, 2$ *in a transformation framework with horizontal and vertical compositionality and decomposition of extension diagrams. Now we have*

$$(COMP_1 \circ_{trafo_1} COMP_2) \circ_{trafo_2} COMP_3 =$$
$$COMP_1 \circ_{trafo_1} (COMP_2 \circ_{trafo_2} COMP_3).$$

Remark: Since the composition of components is only defined for composable components the equality stated above also means that if one composition is defined also the other composition is defined.

$$\begin{tikzcd}
 & & & EXP_1 \arrow[d, Rightarrow, "exp_1"] \\
 & & IMP_1 \arrow[r, "imp_1"] \arrow[d, Rightarrow, "trafo_1"] \arrow[ddr, phantom, "(1)"] & BOD_1 \arrow[dd, Rightarrow, "xtrafo'_1"] \\
 & & EXP_2 \arrow[d, Rightarrow, "exp_2"] & \\
 & IMP_2 \arrow[r, "imp_2"] \arrow[d, Rightarrow, "trafo_2"] \arrow[ddr, phantom, "(3)"] & BOD_2 \arrow[r, "imp'_1"] \arrow[dd, Rightarrow, "xtrafo'_2"] \arrow[ddr, phantom, "(2)"] & BOD_a \arrow[dd, Rightarrow, "xtrafo''_2"] \\
 & EXP_3 \arrow[d, Rightarrow, "exp_3"] & & \\
IMP_3 \arrow[r, "imp_3"] & BOD_3 \arrow[r, "imp'_2"] & BOD_b \arrow[r, "imp''_1"] & BOD_c
\end{tikzcd}$$

Proof: In a first step it is shown that the two composition expressions are defined for the same set of components and transformations.

1. Assumed that the composition $(COMP_1 \circ_{trafo_1} COMP_2) \circ_{trafo_2} COMP_3$ exists. This means that $COMP_1$ is composable with $COMP_2$ via $trafo_1$ and that the composition $COMP_1 \circ_{trafo_1} COMP_2$ is composable with $COMP_3$ via $trafo_2$. So we have the extension diagrams (1) and (3)+(2) above. Using the decomposition of extension diagrams we obtain the extension diagrams (2) and (3). After composing the extension diagram (1)+(2) we have that $COMP_2$ is composable with $COMP_3$ via $trafo_2$ and that $COMP_1$ is composable with $COMP_2 \circ_{trafo_2} COMP_3$ via $trafo_1$.

2. Assumed that the composition $COMP_1 \circ_{trafo_1} (COMP_2 \circ_{trafo_2} COMP_3)$ exists. Then we have the extension diagrams (3) and (1)+(2). Using the decomposition we obtain the extension diagrams (1) and (2). Now we can compose the extension diagram (3)+(2) and obtain that $COMP_1$ is composable with $COMP_2$ via $trafo_1$ and that $COMP_1 \circ_{trafo_1} COMP_2$ is composable with $COMP_3$ via $trafo_2$.

Now we assume the case that both composition expressions are defined. This means that we can construct the extension diagrams (1), (2) and (3) and by that also (1)+(2) and (3)+(2). First we examine the left side of the theorem's equation.

$$\begin{aligned}
& (COMP_1 \circ_{trafo_1} COMP_2) \circ_{trafo_2} COMP_3 \\
= \ & (IMP_2, EXP_1, BOD_a, imp'_1 \circ imp_2, xtrafo'_1 \circ exp_1) \\
& \circ_{trafo_2} COMP_3 \\
& \text{(because of extension diagram (1) and def. “}\circ\text{”)} \\
= \ & (IMP_3, EXP_1, BOD_c, imp''_1 \circ imp'_2 \circ imp_3, xtrafo''_2 \circ xtrafo'_1 \\
& \circ exp_1) \\
& \text{(because of extension diagram (2)+(3) and def. “}\circ\text{”)}
\end{aligned}$$

Now we examine the other side of the theorem's equation.

$$\begin{aligned}
& COMP_1 \circ_{trafo_1} (COMP_2 \circ_{trafo_2} COMP_3) \\
= \ & COMP_1 \circ_{trafo_1} (IMP_3, EXP_2, BOD_b, imp'_2 \circ imp_3, xtrafo'_2 \circ \\
& exp_2) \\
& \text{(because of extension diagram (3) and def. “}\circ\text{”)} \\
= \ & (IMP_3, EXP_1, BOD_c, imp''_1 \circ imp'_2 \circ imp_3, xtrafo''_2 \circ xtrafo'_1 \circ \\
& exp_1) \\
& \text{(because of extension diagram (1)+(2) and def. “}\circ\text{”)}
\end{aligned}$$

This implies that

$$(COMP_1 \circ_{trafo_1} COMP_2) \circ_{trafo_2} COMP_3 =$$
$$COMP_1 \circ_{trafo_1} (COMP_2 \circ_{trafo_2} COMP_3). \quad \square$$

Chapter 3

High-Level Replacement Systems

In this chapter the main definitions and results of high-level replacement systems are introduced. Moreover, several results are shown that are especially important for the abstract component framework introduced in chap. 2. HLR systems as an abstraction of graph transformation systems were introduced in [6]. The main idea was to have a framework that reflects the shared properties of different transformation based specification approaches. Several instantiations have been examined by now, for example: graph transformations, algebraic specifications and different kinds of petri nets. A detailed description of these instantiations is given in [4]. In [3] you can find a discussion of an instantiation of HLR systems to abstract state charts and action nets.

3.1 Introduction to High-Level Replacement Systems

Before defining what a high-level replacement system is, it has to be pointed out that the literature knows several slightly different definitions. For example [24] uses a notion including a class of terminal colored objects. This possible part of an HLR system is not needed in this context and omitted here.

Definition 3.1.1 (HLR System) *An* HLR system $HLRS = (S, P)$ *in a category* **CAT** *is given by an object* S*, called start object and a set* P *of rules.*

Note that sometimes the rules of an HLR system are called *productions*. The definition of the HLR systems left open what these rules look like. They are defined next.

Definition 3.1.2 (HLR Rule) *A* rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ *in* **CAT** *consists of the objects L, K and R, called left-hand side, interface (or gluing object) and right-hand side, and two morphisms $K \overset{l}{\rightarrow} L$ and $K \overset{r}{\rightarrow} R$ belonging to class M.*

The class of morphism $M \subseteq \mathbf{CAT_{mor}}$ is needed as a possibility to constrain what kind of morphisms are allowed in a rule. For some properties of HLR systems such constraints are needed conditions.

The rules are intended to constrain how a particular transformation step is achieved. If we apply a rule to a certain object we obtain a *direct transformation.*

Definition 3.1.3 (Direct Transformation) *Given a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ and an object C, called context object, together with a morphism $K \overset{c}{\rightarrow} C$, a* direct transformation $G_0 \overset{p}{\Rightarrow} G_1$, *from an object G_0 to an object G_1 is given by the following two pushout diagrams in the category* **CAT**.

$$\begin{tikzcd}
L \arrow[d] \arrow[dr, phantom, "(1)"] & K \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "(2)"] & R \arrow[d] \\
G_0 & C \arrow[l] \arrow[r] & G_1
\end{tikzcd}$$

This definition reflects the idea of deleting the image of the left side of the rule in a first step and then adding the image of the right side of the rule. At this point it is important to mention that rules are not always *applicable* to objects. The next definition describes the needed conditions to construct a transformation from a rule.

Definition 3.1.4 (Applicability of Rules) *Given a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ and an object G_0 with a morphism $g : L \rightarrow G_0$ then the rule p is* applicable *via g if:*

1. *there is an object C with morphisms $c : K \rightarrow C$ and $cl : C \rightarrow G_0$ such that square* (1) *in the diagram above becomes a pushout and*

2. *there is an object G_1 with morphisms $h : R \rightarrow G_1$ and $cr : C \rightarrow G_1$ such that* (2) *becomes a pushout square as well.*

Definition 3.1.5 (Transformation) *A* transformation *is defined by a sequence of direct transformations. Such a sequence from an object G_0 to an object G_n is denoted as $G_0 \Rightarrow^* G_n$.*

3.2 High-Level Replacement Systems Satisfy the Embedding Theorem

In this section one important property of HLR systems is formulated and proven. It is of special interest for the component concept, concerning the composition and the semantics of components. These two aspects are discussed in more detail in the sections 2.4 and 2.3. It has to be pointed out that this section only covers one part of the embedding theorem. The classical embedding theorem for graph transformations consists of three parts: the JOIN part and the CLIP part and the bijective correspondence. The theorem 3.2.3 formulated and proven below represents a categorical abstraction of the JOIN part.

For the proofs of theorems in this section three assumptions concerning the underlying category are used.

Assumption 1 *Given three objects A, B, C and morphisms $A \leftarrow B \rightarrow C$, where $B \rightarrow C$ has to be in class M,then there is a pushout $A \rightarrow D \leftarrow C$.*

According to [19] this property is called *existence of semi-M-pushouts.*

Assumption 2 *For each pushout diagram as below, $B \rightarrow C \in M$ implies $A \rightarrow D \in M$.*

$$
\begin{tikzcd}
B \arrow[r] \arrow[d] & C \arrow[d] \\
A \arrow[r] & D
\end{tikzcd}
$$

In [19] this condition is called *inheritance of M-morphisms under pushouts.* Additionally, the following properties of the class of morphisms M are assumed.

Assumption 3 *Given the objects G_1, G_2, G_3, and morphisms $f_1 : G_1 \rightarrow G_2$, $f_1 : G_2 \rightarrow G_3$ either such that $f_1, f_2 \in M$ or such that f_1 is a* **CAT**-*isomorphism and f_2 in M then we have that the morphism $f_2 \circ f_1 : G_1 \rightarrow G_3$ is in class M as well. Moreover, the class M contains all identities.*

These assumptions are well known as a part of HLR conditions. In [19] several conditions for different theorems are listed. But since most of them are needed for concurrency and parallelism theorems here only the needed conditions are examined.

In order to express an adequate notion of consistency for morphisms with respect to HLR transformations we need the following property of pushouts.

Definition 3.2.1 (Initial Pushout) *A pushout $B \stackrel{d}{\to} D \stackrel{c}{\leftarrow} C$ over $B \stackrel{b}{\leftarrow} A \stackrel{a}{\to} C$ is called* initial *with respect to d, if $b \in M$ and for every pushout (1), where $\overline{b}$ has to be a in class M, there are unique morphisms $b' : A \to A'$ and $c' : C \to C'$, such that:*

1. *$b', c' \in M$,*
2. *$b = \overline{b} \circ b'$ and*
3. *$c = \overline{c} \circ c'$ and*
4. *diagram (2) is a pushout.*

$$
\begin{tikzcd}
A \arrow[rr, "b"] \arrow[dr, "b'"] \arrow[ddd, "a"'] & & B \arrow[ddd, "d"] \\
& A' \arrow[ur, "\overline{b}"'] \arrow[d, "a'"] \arrow[dl, phantom, "(2)"] \arrow[dr, phantom, "(1)"] & \\
& C' \arrow[dr, "\overline{c}"] & \\
C \arrow[ur, "c'"] \arrow[rr, "c"'] & & D
\end{tikzcd}
$$

Remark: In the rest of this work the pushout diagram (1) is called *comparative pushout* with respect to the initial pushout. Note that there are already definitions for initial pushouts in [24] and [19] but here it was necessary to add the condition that some of the morphisms are M-morphism because otherwise some of results presented next would not hold.

Theorem 3.2.1 (Uniqueness of Initial Pushouts I) *Given two initial pushouts with respect to the same morphism $d : G_1 \to G_2$ as (1) and (2) in the diagram below. If now assumption 3 holds then we obtain isomorphisms $i_1 : B_1 \to B_2$ and $i_2 : C_1 \to C_2$.*

$$
\begin{tikzcd}
B_1 \arrow[r, "b_1"] \arrow[rr, bend left, "i_1"] \arrow[d, "w_1"] \arrow[dr, phantom, "(1)"] & G_1 \arrow[d, "d"] \arrow[dr, phantom, "(2)"] & B_2 \arrow[l, "b_2"'] \arrow[d, "w_2"] \\
C_1 \arrow[r, "c_1"] \arrow[rr, bend right, "i_2"'] & G_2 & C_2 \arrow[l, "c_2"']
\end{tikzcd}
\qquad
\begin{tikzcd}
B_1 \arrow[r, "i_1"] \arrow[rrr, bend left, "b_1"] \arrow[d, "w_1"] \arrow[dr, phantom, "(4)"] & B_2 \arrow[r, "j_1"] \arrow[d, "w_2"] \arrow[dr, phantom, "(5)"] & B_1 \arrow[r, "b_1"] \arrow[d, "w_1"] \arrow[dr, phantom, "(1)"] & G_1 \arrow[d, "d"] \\
C_1 \arrow[r, "i_2"] \arrow[rrr, bend right, "b_2"'] & C_2 \arrow[r, "j_2"] & C_1 \arrow[r, "c_1"] & G_2
\end{tikzcd}
$$

Proof: Given the initial pushouts (1) and (2) as in the diagram in above. Now we can use the initiality of (1) for the comparative pushout (2). So we obtain M-morphisms $i_1 : B_1 \to B_2$ and $i_2 : C_1 \to C_2$ such that:

$b_1 = b_2 \circ i_1$
$c_1 = c_2 \circ i_2$
(4) is a pushout.

Using the initiality of (2) for the comparative pushout (1) delivers M-morphisms $j_1 : B_2 \to B_1$ and $j_2 : C_2 \to C_1$ such that:

$b_2 = b_1 \circ j_1$
$c_2 = c_1 \circ j_2$.
(5) is a pushout.

We can combine the equations to obtain:

$b_1 = b_1 \circ j_1 \circ i_1$
$b_2 = b_2 \circ i_1 \circ j_1$
$c_1 = c_1 \circ j_2 \circ i_2$
$c_2 = c_2 \circ i_2 \circ j_2$.

If we now use use the initiality of (1) for the same diagram we obtain unique M-morphisms $k_1 : B_1 \to B_1$ and $k_2 : C_1 \to C_1$ such that

$b_1 = b_1 \circ k_1$
$c_1 = c_1 \circ k_2$
and we have a pushout diagram with the morphisms w_1, k_1, k_2.

Since id_{B_1} together with id_{C_1} as well as $j_1 \circ i_1$ together with $j_2 \circ i_2$ have the same properties we obtain

$id_{B_1} = j_1 \circ i_1$
$id_{C_1} = j_2 \circ i_2$.

If we now use use the initiality of (2) for the same diagram we obtain unique M-morphisms $h_1 : B_2 \to B_2$ and $h_2 : C_2 \to C_2$ such that

$b_2 = b_2 \circ h_1$
$c_2 = c_2 \circ h_2$
and we have a pushout diagram with the morphisms w_2, h_1, h_2.

Since id_{B_2} together with id_{C_2} as well as $i_1 \circ j_1$ together with $i_2 \circ j_2$ have the same properties we obtain

$id_{B_2} = i_1 \circ j_1$
$id_{C_2} = i_2 \circ j_2$.

This implies that the constructed morphisms i_1, i_2, j_1, j_2 are isomorphisms. □

Theorem 3.2.2 (Uniqueness of Initial Pushouts II) *Given an initial pushout (1) w.r.t. a morphism d and a pushout (2), where i and i' are supposed to be isomorphisms, then the pushout (2)+(1) is also initial w.r.t. d, if assumption 3 holds.*

$$\begin{tikzcd}
B_2 \arrow[r, "i"] \arrow[d, "w_1"'] \arrow[dr, phantom, "(2)"] & B_1 \arrow[r, "b_1"] \arrow[d, "w_2"'] \arrow[dr, phantom, "(1)"] & G_1 \arrow[d, "d"] \\
C_2 \arrow[r, "i'"] & C_1 \arrow[r, "c"] & G_2
\end{tikzcd}$$

Proof: First we define $b_2 = b_1 \circ i$ and $c_2 = c_1 \circ i'$. Note that b_2 is an M-morphism by assumption 3 because b_1 is an M-morphism by the initiality of (1) and i is an isomorphism. Now given a comparative pushout (C) with respect to the morphism d.

$$\begin{tikzcd}
B_2 \arrow[r, "i"] \arrow[ddd, "w_2"] & B_1 \arrow[rr, "b_1"] \arrow[ddd, "w_1"] \arrow[dr, "\overline{b}"'] & & G_1 \arrow[ddd, "d"] \\
& & B' \arrow[ur, "j_1"'] \arrow[d, "j_2"'] & \\
& \arrow[l, phantom, "(2)"] \arrow[r, phantom, "(3)"] & C' \arrow[dr, "j_3"] \arrow[r, phantom, "(C)"] & \\
C_2 \arrow[r, "i'"] & C_1 \arrow[rr, "c_1"'] \arrow[ur, "\overline{c}"] & & G_2
\end{tikzcd}$$

The initial property of (1) implies that we have unique morphisms $\overline{b}$ and $\overline{c}$ such that:

1. $\overline{b}, \overline{c} \in M$,
2. $b_1 = j_1 \circ \overline{b}$,
3. $c_1 = j_3 \circ \overline{c}$ and
4. (3) is a pushout.

Now we can compose the morphisms $b' = \overline{b} \circ i$ and $c' = \overline{c} \circ i'$. Assumption 3 then implies that b' and c' are in class M. Note that (2)+(3) is a pushout diagram and we have:

$$
\begin{array}{lll}
 & b_1 = j_1 \circ \overline{b} & \text{initiality of (1)} \\
\Rightarrow & b_1 \circ i = j_1 \circ \overline{b} \circ i & \\
\Rightarrow & b_2 = j_1 \circ b' & \text{def. } b_2, b'
\end{array}
$$

$$
\begin{array}{lll}
 & c_1 = j_3 \circ \overline{c} & \text{initiality of (1)} \\
\Rightarrow & c_1 \circ i' = j_3 \circ \overline{c} \circ i' & \\
\Rightarrow & c_2 = j_3 \circ c' & \text{def. } c_2, c'
\end{array}
$$

Now it remains to show that b' and c' are unique w.r.t. the properties shown for them. Assumed there are M-morphisms $\overline{cb} : B_2 \to B'$ and $\overline{cc} : C_2 \to B'$, such that the resulting diagram (3') is a pushout and we have $b_2 = j_1 \circ \overline{cb}$ and $c_2 = j_3 \circ \overline{cc}$.

$$
\begin{tikzcd}
B_2 \arrow[r, "i"] \arrow[d, "w_2\ (2)"] & B_1 \arrow[r, "i^{-1}"] \arrow[d, "w_1\ (2')"] \arrow[rr, bend left, "\overline{b}"] & B_2 \arrow[r, "\overline{bc}"] \arrow[d, "w_2\ (3')"] & B' \arrow[d, "j_2"] \\
C_2 \arrow[r, "i'"] & C_1 \arrow[r, "i'^{-1}"] \arrow[rr, bend right, "\overline{c}"'] & C_2 \arrow[r, "\overline{cc}"] & C'
\end{tikzcd}
$$

Now we have:

$$
\begin{array}{lll}
 & b_2 = j_1 \circ \overline{cb} & \\
\Rightarrow & b_1 \circ i = j_1 \circ \overline{cb} & (\text{def. } b_2) \\
\Rightarrow & b_1 = j_1 \circ \overline{cb} \circ i^{-1} & (i \text{ is isom.})
\end{array}
$$

And we have:

$$
\begin{array}{lll}
 & c_2 = j_3 \circ \overline{cc} & \\
\Rightarrow & c_1 \circ i' = j_3 \circ \overline{cc} & \text{(def. } c_2\text{)} \\
\Rightarrow & c_1 = j_3 \circ \overline{cc} \circ i'^{-1} & (i' \text{ is isom.})
\end{array}
$$

Since we know that diagram (2) is a pushout and diagram (2)+(2') is a pushout because the horizontal morphisms are identities we can conclude that (2') is a pushout as well. So we have M-morphisms $\overline{bc} \circ i^{-1}$ and $\overline{cc} \circ i'^{-1}$ such that $b_1 = j_1 \circ \overline{cb} \circ i^{-1}$ and $c_1 = j_3 \circ \overline{cc} \circ i'^{-1}$ and (2')+(3') is a pushout. Now we can use the uniqueness of $\overline{b}$ and $\overline{c}$ and obtain:

$$
\begin{array}{l}
\overline{b} = \overline{bc} \circ i^{-1} \\
\overline{c} = \overline{cc} \circ i'^{-1}
\end{array}
$$

¿From this equations we can conclude:

$$
\begin{array}{l}
b' = \overline{b} \circ i = \overline{bc} \\
c' = \overline{c} \circ i' = \overline{cc}
\end{array}
$$

We have shown that b' and c' are unique under the given properties. So the pushout (2)+(1) is initial with respect to d. $\square$

Definition 3.2.2 (Consistent Morphisms) *A morphism $k : G_0 \to G_0'$ is* consistent *with respect to a transformation $t : G_0 \Rightarrow^* G_n$, with $t = t_1; \dots ; t_n$, if the following two conditions are satisfied.*

$$
\begin{tikzcd}
Bound \arrow[r, "b"] \arrow[d, "w"] \arrow[rr, dotted, bend left, "j_1"] \arrow[rrrr, dotted, bend left, "j_2"] \arrow[rrrrrr, dotted, bend left, "j_n"] \arrow[dr, phantom, "(1)"] & G_0 \arrow[d, "k"] & C_1 \arrow[l, "cl_1"'] \arrow[r, "cr_1"] & G_1 & C_2 \arrow[l, "cl_2"'] \arrow[r, "cr_2"] & G_2 \arrow[r, dotted, no head] & C_n \arrow[r, "cr_n"] & G_n \\
Con \arrow[r, "con"] & G_0' & & & & & &
\end{tikzcd}
$$

1. *There is an initial pushout with respect to $k : G_0 \to G_0'$, given by diagram (1) above.*

2. *There are morphisms $j_i : Bound \to C_i$, for $i = 1, \dots , n$, such that:*

   (a) $j_i \in M$, for $1 \leq i \leq n$,

   (b) $cl_1 \circ j_1 = b$

(c) $cr_{i-1} \circ j_{i-1} = cl_i \circ j_i$, *for* $1 < i \leq n$,

where n *denotes the number of direct transformations in* t *and* C_i *the image of the interface of the* i*-th direct transformation* t_i *in* t. $cl_i : C_i \to G_{i-1}$ *and* $cr_i : C_i \to G_i$ *denote the corresponding morphisms of* t_i *in* t.

Note that in the diagram above the corresponding rules for the direct transformations $t_{1,\ldots,n}$ *are omitted.*

Definition 3.2.3 (HLR Extension Diagram) *A diagram of two HLR transformations* $t : G_0 \Rightarrow^* G_n$, $t' : G'_0 \Rightarrow^* G'_n$, *where* $t = t_1; \ldots ; t_n$ *and* $t' = t'_1; \ldots ; t'_n$, *and two morphisms* k_0, k_n *as in the picture below is called HLR extension diagram if:*

1. k_0 *is consistent with respect to* t *and*

2. G'_n, t', k_n *can be obtained by applying the extension property on* t, k_0. *This means that we have a sequence of connected double pushouts* (1_i) *and* (2_i) *as depicted below for* $i = 1, \ldots , n$ *such that the diagram with* $Con, Bound, C_i, C'_i$ *is a pushout, where* $Bound$ *and* Con *belong to the boundary construction over* k_0 *and* j_i *is the morphism obtained by the consistency of* k_0 *w.r.t.* t. C_i *and* C'_i *are the images of the interfaces of the* i*-th direct transformation in* t *and* t' *respectively.*

$$\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t"] \arrow[d, "k_0"] & G_n \arrow[d, "k_n"] & Bound \arrow[d, "w"] \arrow[rrr, dotted, bend left, "j_i"] & G_{i-1} \arrow[d, "k_{i-1}"] \arrow[dr, phantom, "(1_i)"] & C_i \arrow[l, "cl_i"'] \arrow[r, "cr_i"] \arrow[d, "c_i"] \arrow[dr, phantom, "(2_i)"] & G_i \arrow[d, "k_i"] \\
G'_0 \arrow[r, Rightarrow, "t'"] & G'_n & Con \arrow[rrr, dotted, bend right, "j'_i"'] & G'_{i-1} & C'_i \arrow[l, "cl'_i"'] \arrow[r, "cr'_i"] & G'_i
\end{tikzcd}$$

Theorem 3.2.3 (Extension Property) *Given a transformation sequence* $t : G_0 \Rightarrow^* G_n$, *with* $t = t_1; \ldots ; t_n$, *and a morphism* $k_0 : G_0 \to G'_0$, *such that* k_0 *is consistent with respect to* t, *then we obtain a selected transformation* $t' : G'_0 \Rightarrow^* G'_n$ *and a selected morphism* $k_n : G_n \to G'_n$, *such that (E) becomes an HLR extension diagram, if assumptions 1 and 2 hold.*

$$\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t"] \arrow[d, "k_0"] \arrow[dr, phantom, "(E)"] & G_n \arrow[d, "k_n"] \\
G'_0 \arrow[r, Rightarrow, "t'"] & G'_n
\end{tikzcd}$$

Remark: In [24] there is a proof for a similar theorem where the consistency condition is expressed in terms of the applicability of extended rules.

Proof: The Proof of this theorem is divided into two steps. Before the validity for sequences of direct transformations is shown, the proof is given for the more simple case of direct transformations.

Direct Transformations Given a direct transformation $G_0 \overset{t}{\Rightarrow} G_1$, consisting of a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$, objects G_0, C, G_1, morphisms c_1, c_2, c_3, cl, cr as depicted below and additionally given a morphism $k : G_0 \rightarrow G_0'$, such that k is consistent with respect to t. (see definition 3.2.2).

$$
\begin{tikzcd}
 & L \arrow[d, "c_1"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "c_2"] & R \arrow[d, "c_3"] \\
Bound \arrow[r, "b"] \arrow[d, "w"] \arrow[rr, dotted, bend left, "j"] \arrow[dr, phantom, "(1)"] & G_0 \arrow[d, "k"] & C \arrow[l, "cl"'] \arrow[r, "cr"] & G_1 \\
Con \arrow[r, "con"] & G_0' & &
\end{tikzcd}
$$

The consistency of k with respect to t implies not only the existence of j but also that $j \in M$. According to assumption 1 we can construct the pushout $Con \rightarrow C' \leftarrow C$ over $Con \overset{w}{\leftarrow} Bound \overset{j}{\rightarrow} C$.

$$
\begin{tikzcd}
 & L \arrow[d, "c_1"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "c_2"] & R \arrow[d, "c_3"] \\
Bound \arrow[r, "b"] \arrow[d, "w"] \arrow[rr, dotted, bend left, "j"] \arrow[dr, phantom, "(1)"] & G_0 \arrow[d, "k"] & C \arrow[l, "cl"'] \arrow[r, "cr"] \arrow[d, "w'"] & G_1 \\
Con \arrow[r, "con"] \arrow[rr, dotted, bend right, "i'"'] & G_0' & C' &
\end{tikzcd}
$$

G_0' is a comparative object with respect to the pushout object C', because we have:

$$
\begin{aligned}
k \circ cl \circ j &= k \circ b && (k \text{ is consistent w.r.t. } p) \\
&= con \circ w && ((1) \text{ is a pushout square})
\end{aligned}
$$

As a result we obtain a unique morphism $cl' : C' \rightarrow G_0'$, such that $cl' \circ i' = con$ and $cl' \circ w' = k \circ cl$.

$$
\begin{tikzcd}
Bound \arrow[r, "j"] \arrow[d, "w"] \arrow[rr, bend left, dotted, "b"] \arrow[dr, phantom, "(1)"] & C \arrow[r, "cl"] \arrow[d, "w'"] \arrow[dr, phantom, "(2)"] & G_0 \arrow[d, "k"] \\
Con \arrow[r, "i'"] \arrow[rr, bend right, dotted, "con"'] & C' \arrow[r, "cl'"] & G'_0
\end{tikzcd}
$$

Because the squares (1) and (1)+(2) are pushouts and (2) commutes we can conclude that (2) is a pushout square as well. Since $r : K \to R$ is in class M and G_1 is constructed as a pushout we can conclude according to assumption 2 that cr is in M as well. According to assumption 1 G'_1 is now constructed as a pushout over $C' \overset{w'}{\leftarrow} C \overset{cr}{\rightarrow} G_1$. Let this pushout be denotated as $C' \overset{cr'}{\rightarrow} G'_1 \overset{w''}{\leftarrow} G_1$. We obtain the following diagram:

$$
\begin{tikzcd}
L \arrow[d, "c_1"] \arrow[dr, phantom, "(a)"] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "c_2"] \arrow[dr, phantom, "(b)"] & R \arrow[d, "c_3"] \\
G_0 \arrow[d, "k"] \arrow[dr, phantom, "(2)"] & C \arrow[l, "cl"'] \arrow[r, "cr"] \arrow[d, "w'"] \arrow[dr, phantom, "(3)"] & G_1 \arrow[d, "w''"] \\
G'_0 & C' \arrow[l, "cl'"'] \arrow[r, "cr'"] & G'_1
\end{tikzcd}
$$

Since all four small squares, $(a), (b), (2), (3)$, are pushouts we can conclude that $(a) + (2)$ and $(b) + (3)$ are pushouts. Now we have a selected direct transformation $G'_0 \overset{t'}{\Rightarrow} G'_1$ based on these two pushouts. And we have a selected morphism $w'' : G_1 \to G'_1$.

Transformations Given a transformation sequence $t : G_0 \Rightarrow^* G_n$, with $t = t_1; \ldots ; t_n$, consisting of rules $p_i = (L_i \overset{l_i}{\leftarrow} K_i \overset{r_i}{\rightarrow} R_{)}$, objects G_i, C, G_i, morphisms $l_i, r_i, cl_i, cr_i, dl_i, dr_i, dk_i$ (for $1 \leq i \leq n$) as depicted below and additionally given a morphism $k : G_0 \to G'_0$, such that k is consistent with respect to t. (see definition 3.2.2).

$$
\begin{tikzcd}
 & L_i \arrow[d, "dl_i"] & K \arrow[l, "l_i"'] \arrow[r, "r_i"] \arrow[d, "dk_i"] & R \arrow[d, "dr_i"] \\
Bound \arrow[d, "w"] \arrow[rr, dotted, bend left, "j_i"] & G_{i-1} \arrow[d, "k_{i-1}"] & C_i \arrow[l, "cl_i"'] \arrow[r, "cr_i"] & G_i \\
Con & G'_{i-1} & &
\end{tikzcd}
$$

Assumed we have an extension up to the $i-1$-th direct transformation in t (for $i \geq 2$). That means we have a transformation $ext_{i-1} : G'_0 \Rightarrow^* G'_{i-1}$ and a morphism $k_{i-1} : G_{i-1} \to G'_{i-1}$ as in the figure above. The consistency of k with respect to t implies that we have $j_i : Bound \to C_i$, such that $j_i \in M$ and $k_{i-1} \circ cl_i \circ j_i = k_{i-1} \circ cr_{i-1} \circ j_{i-1}$. So we can construct the pushout over the morphisms w and j_i.

$$
\begin{tikzcd}
 & L_i \arrow[d, "dl_i"] & K \arrow[l, "l_i"'] \arrow[r, "r_i"] \arrow[d, "dk_i"] & R \arrow[d, "dr_i"] \\
Bound \arrow[d, "w"] \arrow[rr, dotted, bend left, "j_i"] & G_{i-1} \arrow[d, "k_{i-1}"] & C_i \arrow[l, "cl_i"'] \arrow[r, "cr_i"] \arrow[d, "c_i"] & G_i \\
Con \arrow[rr, dotted, bend right, "j'_i"'] & G'_{i-1} & C'_i &
\end{tikzcd}
$$

Now G'_{i-1} is a comparative object with respect to the pushout object C'_i, because we have:

$$
\begin{aligned}
k_{i-1} \circ cl_i \circ j_i &= k_{i-1} \circ cr_{i-1} \circ j_{i-1} && (k \text{ is consistent w.r.t. } p) \\
&= cr_{i-1} \circ j'_{i-1} \circ w && ((1+2) \text{ is a pushout square})
\end{aligned}
$$

$$
\begin{tikzcd}
Bound \arrow[r, "j_{i-1}"] \arrow[d, "w"] \arrow[rrr, dotted, bend left, "j_i"] \arrow[dr, phantom, "(1)"] & C_{i-1} \arrow[r, "cr_{i-1}"] \arrow[d, "c_{i-1}"] \arrow[dr, phantom, "(2)"] & G_{i-1} \arrow[d, "k_{i-1}"] \arrow[dr, phantom, "(3)"] & C_i \arrow[l, "cl_i"'] \arrow[r, "cr_i"] \arrow[d, "c_i"] & G_i \arrow[d, "k_i"] \\
Con \arrow[r, "j'_{i-1}"] \arrow[rrr, dotted, bend right, "j'_i"'] & C'_{i-1} \arrow[r, "cr_{i-1}"] & G'_{i-1} & C'_i \arrow[l, "cl'_i"'] \arrow[r, "cr'_i"] & G'_i
\end{tikzcd}
$$

As a result we obtain a unique morphism $cl'_i : C'_i \to G'_{i-1}$, such that $cl'_i \circ j'_i = cr_{i-1} \circ j'_{i-1}$ and $cl'_i \circ c_i = k_{i-1} \circ cl_i$. By the decompositionality of pushouts we can conclude that (3) is a pushout. By assumption 2 we can conclude that cr_i is in class M, so we can construct G'_i as pushout over the morphisms c_i and cr_i. We also obtain a morphism $k_i : G_i \to G'_i$.

Since all horizontal morphism are in class M we have now a direct transformation $G'_{i-1} \overset{step_i}{\Longrightarrow} G'_i$. Together with the assumed ext_{i-1} we can compose the selected transformation $G'_0 \overset{ext_i}{\Longrightarrow} G'_i$. Together with the morphism k_i we have then an extension up to the i-th direct transformation in t. In the case $i = n$ we can define the wanted extension as $t' = ext_n$. Using the principle of complete induction together with the induction anchor (proof for direct transformations) and the results above we can conclude that the theorem is valid for all sequences of direct transformations. $\square$

Remark: Note that this proof does not use the fact that the pushout (1) is initial. The initiality of (1) is needed to get a stronger correspondence to the extension diagrams, where we need the initial pushouts to obtain the different compositionality results.

3.3 Compositionality of Embedding Diagrams

After introducing and proving the needed part of an embedding theorem in the previous section now two properties concerning the compositionality of embedding diagrams are formulated. They are needed for the composition and the semantics of components. (See sect. 2.4 and sect. 2.3 for details.)

The next lemmas require three additional general assumptions concerning the underlying HLR category **CAT** and the class of morphisms M.

Assumption 4 *M defines a class of monomorphisms in* **CAT**.

Assumption 5 *For each pair of objects $A, B \in$* **CAT** *with M-morphism $d : A \to B$ there is an initial pushout $B \overset{con}{\leftarrow} Con \overset{w}{\leftarrow} Bound \overset{b}{\to} A$ with respect to d.*

$$\begin{tikzcd} Bound \arrow[r, "b"] \arrow[d, "w"] & A \arrow[d, "d"] \\ Con \arrow[r, "con"] & B \end{tikzcd}$$

Note that this assumption corresponds to the existence of a *boundary* and a *context* in the sense of the classical graph transformation approach. According to that initial pushouts constructed over M-morphisms are also called *boundaries*.

Assumption 6 *Given a pushout diagram as below, where $B \to C, C \to D$ and $A \to D$ are supposed to be in M, then we have that $B \to A$ is in M as well.*

$$\begin{tikzcd}
B \arrow[r] \arrow[d] & C \arrow[d] \\
A \arrow[r] & D
\end{tikzcd}$$

Theorem 3.3.1 (Pushouts Preserve Boundaries) *If we have a pushout diagram as (2) below, where cl_1 is supposed to be in M, and assumptions 2, 3, 4 hold and we have the boundaries (1) and (3) over the morphisms d and g respectively then the diagram (2+3) below is an initial pushout with respect to d and we obtain morphisms $a_1 : B_G \to C_1$, $a_2 : C_G \to C_1'$ such that the diagram (Y) is an initial pushout with respect to g and $b_1 = cl_1 \circ a_1$ and $b_2 = cl_1' \circ a_2$.*

$$\begin{tikzcd}
B_G \arrow[r, "b_1"] \arrow[d, "b_G"] \arrow[dr, phantom, "(1)"] & G_0 \arrow[d, "d"] \arrow[dr, phantom, "(2)"] & C_1 \arrow[l, "cl_1"'] \arrow[d, "g"] \arrow[dr, phantom, "(3)"] & B_C \arrow[l, "c_1"'] \arrow[d, "b_C"] \\
C_G \arrow[r, "b_2"] & G_0' & C_1' \arrow[l, "cl_1'"'] & C_C \arrow[l, "c_2"']
\end{tikzcd}
\qquad
\begin{tikzcd}
B_G \arrow[r, "a_1"] \arrow[d, "b_G"] \arrow[dr, phantom, "(Y)"] & C_1 \arrow[d, "g"] \\
C_G \arrow[r, "a_2"] & C_1'
\end{tikzcd}$$

Proof: Given the diagrams (1), (2) and (3) as above, such that (2) is a pushout, cl_1 is in class M, (1) is an initial pushout w.r.t. d and (3) is an initial pushout w.r.t. g. If we use the initial property of (1) for the comparative pushout (2+3) we obtain the M-morphisms e_1, e_2, such that:

- $cl_1 \circ c_1 \circ e_1 = b_1$ (*1a)
- $cl_1' \circ c_2 \circ e_2 = b_2$ (*1b)
- the diagram (X) is a pushout.

$$\begin{tikzcd}
B_G \arrow[r, "b_1"] \arrow[d, "b_G"] \arrow[dr, phantom, "(1)"] \arrow[rrr, bend left, "e_1"] & G_0 \arrow[d, "d"] \arrow[dr, phantom, "(2)"] & C_1 \arrow[l, "cl_1"'] \arrow[d, "g"] \arrow[dr, phantom, "(3)"] & B_C \arrow[l, "c_1"'] \arrow[d, "b_C"] \\
C_G \arrow[r, "b_2"] \arrow[rrr, bend right, "e_2"'] & G_0' & C_1' \arrow[l, "cl_1'"'] & C_C \arrow[l, "c_2"']
\end{tikzcd}
\qquad
\begin{tikzcd}
B_G \arrow[r, "e_1"] \arrow[d, "b_G"] \arrow[dr, phantom, "(X)"] & B_C \arrow[d, "b_C"] \\
C_G \arrow[r, "e_2"] & C_C
\end{tikzcd}$$

Using the initiality of (1) for the comparative pushout (2) delivers a pushout (4) such that:

- $cl_1 \circ f_1 = b_1$ (*2a)
- $cl'_1 \circ f_2 = b_2$ (*2b).

$$\begin{tikzcd}
B_G \arrow[r, "f_1"] \arrow[d, "b_G"] \arrow[rr, bend left, "b_1"] \arrow[dr, phantom, "(4)"] & C_1 \arrow[r, "cl_1"] \arrow[d, "g"] \arrow[dr, phantom, "(2)"] & G_0 \arrow[d, "d"] \\
C_G \arrow[r, "f_2"] \arrow[rr, bend right, "b_2"'] & C'_1 \arrow[r, "cl'_1"] & G'_0
\end{tikzcd}$$

By applying the initiality of pushout (3) on the comparative pushout (4) we obtain a pushout (5) such that h_1 and h_2 are in M and we have:

- $f_1 \circ h_1 = c_1$ (*3a)
- $f_2 \circ h_2 = c_2$ (*3b)

$$\begin{tikzcd}
B_C \arrow[r, "h_1"] \arrow[d, "b_C"] \arrow[rr, bend left, "c_1"] \arrow[dr, phantom, "(5)"] & B_G \arrow[r, "f_1"] \arrow[d, "b_G"] \arrow[dr, phantom, "(4)"] & C_1 \arrow[d, "g"] \\
C_C \arrow[r, "h_2"] \arrow[rr, bend right, "c_2"'] & C_G \arrow[r, "f_2"] & C'_1
\end{tikzcd}$$

In order to show that e_1, h_1, e_2 and h_2 are isomorphisms it is shown next that $b_1 = b_1 \circ h_1 \circ e_1$ and $b_2 = b_2 \circ h_2 \circ e_2$.

$$\begin{aligned}
b_1 \circ h_1 \circ e_1 &= cl_1 \circ f_1 \circ h_1 \circ e_1 && \text{(because *2a)} \\
&= cl_1 \circ c_1 \circ e_1 && \text{(because *3a)} \\
&= b_1 && \text{(because *1a)}
\end{aligned}$$

$$\begin{aligned}
b_2 \circ h_2 \circ e_2 &= cl'_1 \circ f_2 \circ h_2 \circ e_2 && \text{(because *2b)} \\
&= cl'_1 \circ c_2 \circ e_2 && \text{(because *3b)} \\
&= b_2 && \text{(because *1b)}
\end{aligned}$$

Since b_1 is an M-morphism and assumption 2 was made b_2 is an M-morphism as well. By assumption 4 we have now that b_1 and b_2 are monomorphisms. Both properties are used next.

$$\begin{aligned}
&b_1 = b_1 \circ id_{B_G} = b_1 \circ h_1 \circ e_1 \\
&\Rightarrow h_1 \circ e_1 = id_{B_G}
\end{aligned}$$

$$\begin{aligned}
&b_2 = b_2 \circ id_{C_G} = b_2 \circ h_2 \circ e_2 \\
&\Rightarrow h_2 \circ e_2 = id_{C_G}
\end{aligned}$$

Now the other direction of the isomorphism property is shown:

$$\begin{aligned}
cl_1 \circ c_1 \circ e_1 \circ h_1 &= b_1 \circ h_1 && \text{(because *1a) (*4a)} \\
&= cl_1 \circ f_1 \circ h_1 && \text{(because *2a)} \\
&= cl_1 \circ c_1 && \text{(because *3a) (*4b)}
\end{aligned}$$

$$\begin{aligned}
cl'_1 \circ c_2 \circ e_2 \circ h_2 &= b_2 \circ h_2 && \text{(because *1b) (*5a)} \\
&= cl'_1 \circ f_2 \circ h_2 && \text{(because *2b)} \\
&= cl'_1 \circ c_2 && \text{(because *3b) (*5b)}
\end{aligned}$$

Per assumption cl_1 is in class M. Moreover, c_1 is an M-morphism because of the initial pushout property of (3). Assumption 2 then implies that cl'_1 and c_2 are M-morphisms as well. According to assumption 4 we have that cl_1, c_1, cl'_1, c_2 are monomorphisms. So $cl_1 \circ c_1$ and $cl'_1 \circ c_2$ are monomorphisms as well.

$$\begin{aligned}
&cl_1 \circ c_1 = cl_1 \circ c_1 \circ id_{B_C} = cl_1 \circ c_1 \circ e_1 \circ h_1 \\
&\Rightarrow e_1 \circ h_1 = id_{B_C}
\end{aligned}$$

$$\begin{aligned}
&cl'_1 \circ c_2 = cl'_1 \circ c_2 \circ id_{C_C} = cl'_1 \circ c_2 \circ e_2 \circ h_2 \\
&\Rightarrow e_2 \circ h_2 = id_{C_C}
\end{aligned}$$

Now it is proven that e_1, h_1, e_2 and h_2 are isomorphisms. By combining (*4a), (*4b) and (*5a) and (*5b) we obtain that:

$$b_1 \circ h_1 = cl_1 \circ c_1 \quad \text{and} \quad b_2 \circ h_2 = cl'_1 \circ c_2. \qquad \text{(*6)}$$

Since we know that e_1, e_2 and h_1, h_2 are in M we can apply theorem 3.2.2 on the diagrams (5)+(1) and (X)+(3) and obtain that (5)+(1) is an initial pushout with respect to d and that (X)+(3) is an initial pushout with respect to g. Since the horizontal morphisms of (5)+(1) are equal to the horizontal morphisms of (2)+(3) by (*6) we have that (2)+(3) is an initial pushout w.r.t. d. And by defining $a_1 = c_1 \circ e_1$ and $a_2 = c_1 \circ e_2$ we obtain that diagram (Y)=(X)+(3) what implies that (Y) is an initial pushout w.r.t. g. Together with the equations (*1a) and (*1b) we can conclude that $b_1 = cl_1 \circ a_1$ and $b_2 = cl'_1 \circ a_2$. $\square$

Theorem 3.3.2 (Double Pushout Sequences Preserve Boundaries) *Given a sequence of double pushouts and an initial pushout (1) as depicted below, where cl_i and cr_i are supposed to be in class M for $1 \leq i \leq n$ and d_0 is supposed to be in M as well. If assumptions 2, 3, 4, 5 and 6 hold then we have morphisms $j_i : B \to C_i$ for $i = 1, \ldots, n$ such that:*

1. *$j_i \in M$, for $1 \leq i \leq n$,*
2. *$cl_1 \circ j_1 = b$*
3. *$cr_{i-1} \circ j_{i-1} = cl_i \circ j_i$, for $1 < i \leq n$.*
4. *C'_i, c_i and a suiting morphism j'_i are a pushout over the morphisms b_G and j_i for $1 \leq i \leq n$.*

$$\begin{tikzcd}
B \arrow[r, "b"] \arrow[d, "b_G"'] \arrow[dr, phantom, "(1)"] & G_0 \arrow[d, "d_0"] & C_1 \arrow[l, "cl_1"'] \arrow[r, "cr_1"] \arrow[d, "c_1"] & G_1 \arrow[d, "d_1"] \arrow[r, dotted, no head] & C_n \arrow[r, "cr_n"] \arrow[d, "c_n"] & G_n \arrow[d, "d_n"] \\
C \arrow[r, "con"] & G'_0 & C'_1 \arrow[l, "cl'_1"'] \arrow[r, "cr'_1"] & G'_1 \arrow[r, dotted, no head] & C'_n \arrow[r, "cr'_n"] & G'_n
\end{tikzcd}$$

Proof: Given the sequence of double pushouts as mentioned above and the initial pushout (1) such that cl_i and cr_i are in class M for $1 \leq i \leq n$ and d_0 is in class M. By assumption 2 we can conclude that cl'_i and cr'_i are in class M for $1 \leq i \leq n$ as well. Starting at d_0 we can now apply assumptions 6 and 2 alternately on the sequence and obtain that the morphisms c_i and d_i are in class M for $1 \leq i \leq n$. According to assumption 5 we can now construct all the boundaries $B_{Ci}, C_{Ci}, B_{Gi}, C_{Gi}$ over the vertical morphisms c_i and d_i for $i = 1, \ldots, n$.

$$\begin{tikzcd}
B_{Ci} \arrow[r, "b_{ci}"] \arrow[d, "w_{ci}"] & C_i \arrow[d, "ci"] & B_{Gi} \arrow[r, "b_{gi}"] \arrow[d, "w_{gi}"] & G_i \arrow[d, "gi"] \\
C_{Ci} \arrow[r, "b'_{ci}"] & C'_i & C_{Gi} \arrow[r, "b'_{gi}"] & G'_i
\end{tikzcd}$$

Assumed we have morphisms $z_i : B \to G_i$ and $z'_i : C \to G'_i$ with $0 \leq i < n$ such that the diagram (P_{i-1}) below is an initial pushout w.r.t. d_i.

$$
\begin{tikzcd}
B \arrow[r, "z_{i-1}"] \arrow[d, "b_G"'] \arrow[rr, bend left, "j_i"] \arrow[rrr, bend left=40, "z_i"] \arrow[dr, phantom, "(P_{i-1})"] & G_{i-1} \arrow[d, "d_{i-1}"] \arrow[dr, phantom, "(O_i)"] & C_i \arrow[l, "cl_i"'] \arrow[r, "cr_i"] \arrow[d, "c_i"] \arrow[dr, phantom, "(U_i)"] & G_i \arrow[d, "d_i"] \\
C \arrow[r, "z'_{i-1}"] \arrow[rr, bend right, "j'_i"'] \arrow[rrr, bend right=40, "z'_i"'] & G'_{i-1} & C'_i \arrow[l, "cl'_i"'] \arrow[r, "cr'_i"] & G'_i
\end{tikzcd}
\qquad
\begin{tikzcd}
B \arrow[r, "j_i"] \arrow[d, "b_G"'] \arrow[dr, phantom, "(F_i)"] & C_i \arrow[d, "c_i"] \\
C \arrow[r, "j'_i"] & C'_i
\end{tikzcd}
$$

Now the theorem 3.3.1 delivers morphisms j_i and j'_i such that the constructed diagram (F_i) is an initial pushout with respect to c_i. Moreover, we have $z_{i-1} = cl_i \circ j_i$ (*1a) and $z'_{i-1} = cl'_i \circ j'_i$ (*1b). We can use theorem 3.3.1 again to obtain that the diagram (F_i)+(U_i) is an initial pushout w.r.t. d_i. So we can define the morphisms $z_i = cr_i \circ j_i$ (*2a) and $z'_i = cr'_i \circ j'_i$ (*2b). So we have constructed an initial pushout (P_i) from the given (P_{i-1}). If we now define (P_0)=(1) we obtain initial pushouts (P_i) w.r.t. d_i for $i = 0, \dots, n$. This corresponds to the first and the fourth statement of the theorem.

Since the equations (*1a) and (*2a) are now valid for $1 \leq i \leq n$ we obtain that $cr_{i-1} \circ j_{i-1} = cl_i \circ j_i$ for $1 < i \leq n$. Since we have $z_0 = b$ we also obtain $b = cl_1 \circ j_1$. This corresponds to the second and the third statement of the theorem. $\square$

Now having these three theorems on the isomorphism of boundaries it is possible to prove the needed compositionality theorems for HLR extension diagrams.

Theorem 3.3.3 (Hor. Compositionality of Extension Diagrams) *Given two HLR extension diagrams like (1) and (2) below where $t_1 : G_0 \Rightarrow^* G_n$, $t_2 : G_n \Rightarrow^* G_m$, $t'_1 : G'_0 \Rightarrow^* G'_n$, $t'_2 : G_n \Rightarrow^* G'_m$ are sequences of direct transformations that consist of direct transformations $t_1 = t_{11}; \dots ; t_{1n}$, $t'_1 = t'_{11}; \dots ; t'_{1n}$, $t_2 = t_{2n}; \dots ; t_{2m}$, $t'_2 = t'_{2n}; \dots ; t'_{2m}$ with $m > n$. This means that t_1 and t'_1 consist of n direct transformations and t_2 and t'_2 consist of $m - n$ direct transformations. Moreover we assume that d_0 is an M-morphism and that assumptions 2, 3, 4, 5, 6 hold. Then the composition of the diagrams (1)+(2), defined by the morphisms d_0 and d_m and the transformations $t_1; t_2 : G_0 \Rightarrow^* G_m$ and $t'_1; t'_2 : G'_0 \Rightarrow^* G'_m$, is also an HLR* *extension diagram.*

$$\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t_1"] \arrow[d, "d_0"] \arrow[dr, phantom, "(1)"] & G_n \arrow[r, Rightarrow, "t_2"] \arrow[d, "d_n"] \arrow[dr, phantom, "(2)"] & G_m \arrow[d, "d_m"] \\
G'_0 \arrow[r, Rightarrow, "t'_1"] & G'_n \arrow[r, Rightarrow, "t'_2"] & G'_m
\end{tikzcd}$$

Proof: By the definition of HLR extension diagrams we obtain the existence of the following sequence of double pushouts and that d_0 is consistent with respect to t_1 what implies the existence of the initial pushout (i1). Since t_1 and t_2 are transformations we can apply assumption 2 and obtain that cl_i and cr_i are in M for $1 \leq i \leq m$.

$$\begin{tikzcd}
B \arrow[r, "b"] \arrow[d, "b_G"] \arrow[dr, phantom, "(i1)"] & G_0 \arrow[d, "d_0"] & C_1 \arrow[l, "cl_1"'] \arrow[r, "cr_1"] \arrow[d, "c_1"] & G_1 \arrow[d, "d_1"] \arrow[r, dotted, no head] & C_n \arrow[r, "cr_n"] \arrow[d, "c_n"] & G_n \arrow[d, "d_n"] & C_{n+1} \cdots \arrow[l, "cl_{n+1}"'] \arrow[d, "c_{n+1}"] \\
C \arrow[r, "con"] & G'_0 & C'_1 \arrow[l, "cl'_1"'] \arrow[r, "cr'_1"] & G'_1 \arrow[r, dotted, no head] & C'_n \arrow[r, "cr'_n"] & G'_n & C'_{n+1} \cdots \arrow[l, "cl'_{n+1}"']
\end{tikzcd}$$

At this point we can apply theorem 3.3.2 on the sequence above. As a result we obtain $j_i : B \to C_i$ for $i = 1, \ldots, m$ such that:

1. $j_i \in M$, for $1 \leq i \leq m$,
2. $cl_1 \circ j_1 = b$
3. $cr_{i-1} \circ j_{i-1} = cl_i \circ j_i$, for $1 < i \leq m$.
4. C'_i, c_i and a suiting morphism j'_i are a pushout over the morphisms b_G and j_i for $1 \leq i \leq m$.

Points one to three imply that d_0 is consistent with respect to the transformation $t_1; t_2$. Together with the existence of the double pushout sequence above and point four this implies that the composed diagram (1)+(2) is an HLR extension diagram. □

Theorem 3.3.4 (Ver. Compositionality of Extension Diagrams) *Given two HLR extension diagrams like (1) and (2) below where $t : G_0 \Rightarrow^* G_z$, $t' : G'_0 \Rightarrow^* G'_z$, $t'' : G''_0 \Rightarrow^* G''_z$ are sequences of direct transformations that consist of direct transformations $t = t_1; \ldots; t_z$, $t' = t'_1; \ldots; t'_z$, $t'' = t''_1; \ldots; t''_z$. We assume d_0 and d'_0 to be M-morphisms. If assumptions 2, 3, 4, 5, 6 hold then the composition of the diagrams (1)+(2), defined by* *the morphisms $d'_0 \circ d_0$, $d'_z \circ d_z$ and the transformations t, t'' is also an HLR extension diagram.*

$$\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t"] \arrow[d, "d_0"] & G_z \arrow[d, "d_z"] \\
G'_0 \arrow[r, Rightarrow, "t'"] \arrow[d, "d'_0"] & G'_z \arrow[d, "d'_z"] \\
G''_0 \arrow[r, Rightarrow, "t''"] & G''_z
\end{tikzcd}$$

Proof: By the definition of HLR extension diagrams we obtain two sequences of double pushouts as below. If we compose the pushouts vertically we obtain a connected sequence of double pushouts, where the morphisms cl_i and cr_i are in M for $1 \leq i \leq n$ because t is a transformation and we have assumption 2. Since d_0 and d'_0 are in M we know by assumption 3 that also $d'_0 \circ d_0$ is an M-morphism. Now we can construct the boundary (i1) over the composition $d'_0 \circ d_0$.

$$\begin{tikzcd}
B \arrow[r, "b"] \arrow[dd, "b_G\,(i1)"] & G_0 \arrow[r, dotted, no head] \arrow[d, "d_0"] & G_{n-1} \arrow[d, "d_{n-1}"] & C_n \arrow[l, "cl_n"'] \arrow[r, "cr_n"] \arrow[d, "c_n"] & G_n \arrow[d, "d_n"] \\
& G'_0 \arrow[r, dotted, no head] \arrow[d, "d'_0"] & G'_{n-1} \arrow[d, "d'_{n-1}"] & C'_n \arrow[l, "cl'_n"'] \arrow[r, "cr'_n"] \arrow[d, "c'_n"] & G'_n \arrow[d, "d'_n"] \\
C \arrow[r, "con"] & G''_0 \arrow[r, dotted, no head] & G''_{n-1} & C''_n \arrow[l, "cl''_n"'] \arrow[r, "cr''_n"] & G''_n
\end{tikzcd}$$

Now we can apply theorem 3.3.2 and obtain morphisms $j_i : B \to C_i$ and $j'_i : C \to C''_i$ for $i = 1, \ldots, n$ such that $d'_0 \circ d_0$ becomes consistent w.r.t. t and we have pushouts as depicted below for $i = 1, \ldots, n$.

$$\begin{tikzcd}
B \arrow[r, "j_i"] \arrow[d, "b_G"] & C_i \arrow[d, "c'_i \circ c_i"] \\
C \arrow[r, "j'_i"] & C''_i
\end{tikzcd}$$

So the composed diagram (1)+(2) is an extension diagram. $\square$

3.4 Decomposition of Extension Diagrams

This section introduces theorems for the vertical and horizontal decomposition of HLR extension diagrams. This is used for the compositionality of the transformation semantics in section 2.5. Note that the theorem 3.4.3 below does not directly match to the CLIP part of the classical embedding theorem because we use tighter conditions for the existence of the corresponding transformation. Unfortunately, it is necessary to formulate new assumptions for the underlying HLR category to show that the decomposition theorems are valid.

Assumption 7 *Given objects B, C, D with M-morphisms $b : B \to D, c : C \to D$. Now we have an object A with morphisms $a_1 : A \to C$ and $a_2 : A \to B$ such that the diagram below becomes a pullback diagram.*

$$\begin{tikzcd} A \arrow[r, "a_2"] \arrow[d, "a_1"'] \arrow[dr, phantom, "(P)"] & B \arrow[d, "b"] \\ C \arrow[r, "c"] & D \end{tikzcd}$$

In [19] this property is called *existence of M-pullbacks.*

Assumption 8 *Given a pullback diagram as (P) above. Now we have if c and b are in class M then also a_1 and a_2 are in class M.*

In [19] this property is called *inheritance of M-morphisms under pullbacks.*

Assumption 9 *If in the diagram (P) above (D, c, b) is a pushout of (A, a_1, a_2) and a_1, a_2 are M-morphisms then (A, a_1, a_2) is a pullback of (D, c, b).*

Assumption 10 *Given a pushout (1)+(2) as depicted below, where (2) is assumed to be a pullback and a_1, b_1, e, b_2, d are assumed to be M-morphisms. Now we have that (1) is a pushout.*

$$\begin{tikzcd} A \arrow[r, "a_2"] \arrow[d, "a_1"'] \arrow[dr, phantom, "(1)"] & B \arrow[r, "b_2"] \arrow[d, "b_1"] \arrow[dr, phantom, "(2)"] & E \arrow[d, "e"] \\ C \arrow[r, "c"] & D \arrow[r, "d"] & F \end{tikzcd}$$

In [19] this property is called *M-pushout-pullback decomposition property.*

Theorem 3.4.1 (Hor. Decomposition of Extension Diagrams) *If we have the HLR extension diagram (1) below where d_0 has to be in class M and $t_1 : G_0 \Rightarrow^* G_n$, $t_2 : G_n \Rightarrow^* G_m$ and $t' : G'_0 \Rightarrow^* G'_m$ are sequences of direct transformations that consist of direct transformations $t_1 = t_{11}; \ldots ; t_{1n}$, $t_2 = t_{2n}; \ldots ; t_{2m}$, $t' = t'_1; \ldots ; t'_m$ with $m > n$ and if the assumptions 2, 3, 4, 5, 6 hold then we obtain the HLR extension diagrams (2) and (3).*

$$\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t_1"] \arrow[d, "d_0"] & G_n \arrow[r, Rightarrow, "t_2"] \arrow[d, phantom, "(1)"] & G_m \arrow[d, "d_m"] \\
G'_0 \arrow[rr, Rightarrow, "t'"] & & G'_m
\end{tikzcd}
\qquad
\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t_1"] \arrow[d, "d_0"] \arrow[dr, phantom, "(2)"] & G_n \arrow[r, Rightarrow, "t_2"] \arrow[d, "d_n"] \arrow[dr, phantom, "(3)"] & G_m \arrow[d, "d_m"] \\
G'_0 \arrow[r, Rightarrow, "t'_1"] & G'_n \arrow[r, Rightarrow, "t'_2"] & G'_m
\end{tikzcd}$$

Proof:

$$\begin{tikzcd}
B \arrow[r, "b"] \arrow[d, "b_G"] \arrow[dr, phantom, "(i1)"] & G_0 \arrow[d, "d_0"] & C_1 \arrow[l, "cl_1"'] \arrow[r, "cr_1"] \arrow[d, "c_1"] & G_1 \arrow[d, "d_1"] \arrow[r, dotted, no head] & C_n \arrow[r, "cr_n"] \arrow[d, "c_n"] & G_n \arrow[d, "d_n"] & C_{n+1} \cdots \arrow[l, "cl_{n+1}"'] \arrow[d, "c_{n+1}"] \\
C \arrow[r, "con"] & G'_0 & C'_1 \arrow[l, "cl'_1"'] \arrow[r, "cr'_1"] & G'_1 \arrow[r, dotted, no head] & C'_n \arrow[r, "cr'_n"] & G'_n & C'_{n+1} \cdots \arrow[l, "cl'_{n+1}"']
\end{tikzcd}$$

Since (1) is an HLR extension diagram we obtain the initial pushout (i1) and the following sequence of double pushouts and morphisms $j_i : B \to C_i$ for $i = 1, \ldots , m$ such that:

1. $j_i \in M$, for $1 \leq i \leq m$,
2. $cl_1 \circ j_1 = b$
3. $cr_{i-1} \circ j_{i-1} = cl_i \circ j_i$, for $1 < i \leq m$.
4. C'_i, c_i and a suiting morphism j'_i are a pushout over the morphisms b_G and j_i for $1 \leq i \leq m$.

Since we have that $n < m$ we have that (2) is an HLR extension diagram because we already have the needed morphisms and pushouts. Moreover, we know by assumption 2 and by the fact that $t_1; t_2$ is a transformation that cl_i, cr_i and cl'_i, cr'_i are in M for $1 \leq i \leq m$. Since d_0 is in M we can use assumptions 6 and 2 alternately on the sequence from the left to the right. So we obtain that the morphisms c_i and d_i are in the class M as well for $1 \leq i \leq m$. As a result we can construct an initial pushout w.r.t. d_n according to assumption 5. Now we can apply theorem 3.3.2 on this initial pushout and the rest of the double pushout sequence and obtain that (3) is an HLR extension diagram. $\square$

Theorem 3.4.2 (Special Pushout Decomposition) *Given the pushout (1) below, where the morphisms d_1, d_2, c, b are supposed to be M-morphisms and given the assumptions 6, 7, 8, 9, 10, then we have an object B with morphisms b_1, b_2, b_3 such that (2) and (3) become pushouts.*

$$
\begin{tikzcd}
G_1 \arrow[d, "d_1"] & A \arrow[l, "a"'] \arrow[dd, "b"] \\
G_2 \arrow[d, "d_2"] \arrow[r, phantom, "(1)"] & \\
G_3 & C \arrow[l, "c"]
\end{tikzcd}
\qquad
\begin{tikzcd}
G_1 \arrow[d, "d_1"] \arrow[r, phantom, "(2)"] & A \arrow[l, "a"'] \arrow[d, "b_1"] \\
G_2 \arrow[d, "d_2"] \arrow[r, phantom, "(3)"] & B \arrow[l, "b_3"] \arrow[d, "b_2"] \\
G_3 & C \arrow[l, "c"]
\end{tikzcd}
$$

Proof: Given the pushout diagram (1) above such that the morphisms d_1, d_2, c, b are M-morphisms. As a first step we can construct (B, b_2, b_3) as a pullback of (G_3, c, d_2). This is possible since we made assumption 7. According to assumption 8 we obtain that b_3 and b_2 are in class M. So we can use assumption 7 again to construct the pullback A', a', b'_1.

$$
\begin{tikzcd}
& & A \arrow[dll, bend right, "a"'] \arrow[dl, "i"] \arrow[ddl, bend left, "b_1"] \\
G_1 \arrow[d, "d_1"] \arrow[r, phantom, "(2')"] & A' \arrow[l, "a'"] \arrow[d, "b'_1"] & \\
G_2 & B \arrow[l, "b_3"] &
\end{tikzcd}
$$

Now we can compose the pullback (3)+(2'). Since c is in M and we have assumption 6 we know that a is in class M. According to assumption 9 we have now that (1) is a pullback over the same morphisms as (3)+(2'). Since pullbacks are unique up to isomorphism we have an isomorphism $i : A \to A'$ such that $b = b_2 \circ b'_1 \circ i$ and $a = a' \circ i$. Now we can define $b_1 = b'_1 \circ i$ and then the uniqueness of pullbacks up to isomorphism implies that (2) is a pullback as well. This implies according to assumption 8 that b_1 is in M. Since we have that $b_1 = b'_1 \circ i$ and $b = b_2 \circ b'_1 \circ i$ we know that $b = b_2 \circ b_1$. So we have that (1)=(2)+(3) and so we know that (2)+(3) is a pushout.

At this point we can apply assumption 10 on the diagrams (2) and (3) because we have that (2)+(3) is a pushout and (3) is a pullback and all morphisms are in M. As a result we obtain that (2) is a pushout. Finally we obtain by the decompositionality of pushouts that (3) is a pushout as well. $\square$

Theorem 3.4.3 (Vertical Decomposition of Extension Diagrams) *Given the extension diagram (1) such that d_0 and d'_0 are in M and given that assumptions 2, 3, 4, 5, 6, 7, 8, 9, 10 hold. Then we have the extension diagrams (2) and (3). Here, $t : G_0 \Rightarrow^* G_n$, $t' : G'_0 \Rightarrow^* G'_n$, $t'' : G''_0 \Rightarrow^* G''_n$ are sequences of direct transformations $t = t_1; \dots ; t_n$, $t' = t'_1; \dots ; t'_n$, $t'' = t''_1; \dots ; t''_n$.*

$$\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t"] \arrow[d, "d_0"] & G_n \arrow[dd, "k_n"] \\
G'_0 \arrow[d, "d'_0"] \arrow[r, phantom, "(1)"] & \\
G''_0 \arrow[r, Rightarrow, "t''"] & G''_n
\end{tikzcd}
\qquad
\begin{tikzcd}
G_0 \arrow[r, Rightarrow, "t"] \arrow[d, "d_0"] \arrow[dr, phantom, "(2)"] & G_n \arrow[d, "d_n"] \\
G'_0 \arrow[r, Rightarrow, "t'"] \arrow[d, "d'_0"] \arrow[dr, phantom, "(3)"] & G'_n \arrow[d, "d'_n"] \\
G''_0 \arrow[r, Rightarrow, "t''"] & G''_n
\end{tikzcd}$$

Proof: Given an HLR extension diagram (1) as above, where d_0, d'_0 are assumed to be in M. By the definition of HLR extension diagrams we obtain the following pushouts for $i = 1, \dots , n$, where $\overline{d_0} = d'_0 \circ d_0$.

$$\begin{tikzcd}
G_0 \arrow[d, "d_0"] & C_1 \arrow[l, "cl_1"'] \arrow[dd, "\overline{c_1}"] \\
G'_0 \arrow[d, "d'_0"] \arrow[r, phantom, "(P_{l1})"] & \\
G''_0 & C''_1 \arrow[l, "cl''_1"']
\end{tikzcd}
\qquad
\begin{tikzcd}
G_{i-1} \arrow[d, "\overline{d_{i-1}}"'] \arrow[dr, phantom, "(P_{li})"] & C_i \arrow[l, "cl_i"'] \arrow[r, "cr_i"] \arrow[d, "\overline{c_i}"] \arrow[dr, phantom, "(P_{ri})"] & G_i \arrow[d, "\overline{d_i}"] \\
G''_{i-1} & C''_i \arrow[l, "cl''_i"'] \arrow[r, "cr''_i"] & G''_i
\end{tikzcd}$$

Now we assume that in a double pushout diagram as below with $1 \leq z \leq n$ we have morphisms $d_{z-1}, d'_{z-1} \in M$ as below such that $\overline{d_{z-1}} = d'_{z-1} \circ d_{z-1}$. On this way the diagram belongs to the sequence of double pushout diagrams

P_{li}, P_{ri} as introduced above.

$$
\begin{tikzcd}
G_{z-1} \arrow[d, "d_{z-1}"] & C_z \arrow[l, "cl_z"'] \arrow[r, "cr_z"] \arrow[dd, "\overline{c_z}"] & G_z \arrow[dd, "\overline{d_z}"] \\
G'_{z-1} \arrow[d, "d'_{z-1}"] \arrow[r, phantom, "(4)"] & \arrow[r, phantom, "(5)"] & {} \\
G''_{z-1} & C''_z \arrow[l, "cl''_z"] \arrow[r, "cr''_z"'] & G''_z
\end{tikzcd}
\qquad
\begin{tikzcd}
G_{z-1} \arrow[d, "d_{z-1}"'] \arrow[dr, phantom, "(4a)"] & C_z \arrow[l, "cl_z"'] \arrow[r, "cr_z"] \arrow[d, "c_z"] \arrow[dr, phantom, "(5a)"] & G_z \arrow[d, "d_z"] \\
G'_{z-1} \arrow[d, "d'_{z-1}"'] \arrow[dr, phantom, "(4b)"] & C'_z \arrow[l, "cl'_z"'] \arrow[r, "cr'_z"] \arrow[d, "c'_z"] \arrow[dr, phantom, "(5b)"] & G'_z \arrow[d, "d'_z"] \\
G''_{z-1} & C''_z \arrow[l, "cl''_z"'] \arrow[r, "cr''_z"] & G''_z
\end{tikzcd}
$$

The definition of transformations together with assumption 2 implies that cl_z and cl''_z are in class M. Since we know that d_0, d''_0 are M-morphisms and we made assumption 6 also $\overline{c_z}$ has to be an M-morphism. At this point we can apply theorem 3.4.2 on the pushout (4) and obtain the pushouts (4a) and (4b). Now assumption 2 implies that cl'_z is a M-morphism. Since cl_z, cl''_z, d_{z-1}, d'_{z-1} are M-morphisms we can conclude by assumption 6 that c_z and c'_z are M-morphisms. Note that theorem 3.4.2 also implies that $c'_z \circ c_z = \overline{c_z}$. Now we can construct the pushouts (5a) and (5b'). By composing the pushouts (5a) and (5b') we obtain a pushout over the same morphisms that (5) was constructed over. The uniqueness of pushouts now delivers an isomorphism $i\ H :\to G''_z$ such that $cr''_z = i \circ h_2$ and $\overline{d_z} = i \circ h_1 \circ d_z$.

$$
\begin{tikzcd}
C'_z \arrow[r, "cr'_z"] \arrow[d, "c'_z"] \arrow[dr, phantom, "(5b')"] & G'_z \arrow[d, "h_1"] \arrow[ddr, bend left, "d'_z"] & \\
C''_z \arrow[r, "h_2"] \arrow[drr, bend right, "cr''_z"'] & H \arrow[dr, "i"] & \\
& & G''_z
\end{tikzcd}
$$

After defining $d'_z = h_1 \circ i$ we can use the uniqueness of pushouts and obtain that (5b) is a pushout. Moreover we have $\overline{d_z} = i \circ h_1 \circ d_z$ and $d'_z = h_1 \circ i$ and so $\overline{d_z} = d'_z \circ d_z$. Assumption 2 applied to pushouts (5a) and (5b) implies that d_z and d'_z are M-morphisms. By choosing $z = 1, \ldots, n$ we obtain the needed context objects C_i and the needed objects G_i together with the corresponding morphisms for $1 \leq i \leq n$ because the assumptions on the morphisms d_z, d'_z are valid in the case $z = 1$ and for each step we obtain the validity for $z+1$.

$$\begin{tikzcd}
B_1 \arrow[d, "w_1"'] \arrow[dr, "b_1"] & & & \\
Con_1 \arrow[dr, "c_1"'] \arrow[r, phantom, "(i1)"] & G_0 \arrow[d, "d_0"] & C_1 \arrow[l, "cl_1"'] \arrow[d, "c_1"] \arrow[r, dotted, no head] & G_n \arrow[d, "d_n"] \\
& G'_0 \arrow[d, "d'_0"] & C'_1 \arrow[l, "cl'_1"'] \arrow[d, "c'_1"] \arrow[r, dotted, no head] & G'_n \arrow[d, "d'_n"] \\
B_2 \arrow[d, "w_2"'] \arrow[ur, "b_2"] \arrow[r, phantom, "(i2)"] & G''_0 & C''_1 \arrow[l, "cl''_1"'] \arrow[r, dotted, no head] & G''_n \\
Con_2 \arrow[ur, "c_2"'] & & &
\end{tikzcd}$$

According to assumption 5 we can construct the initial pushouts (i1) and (i2). Now we can apply theorem 3.3.2 twice and obtain morphisms $j_i : B_1 \to C_i$ and $g_i : B_2 \to C'_i$ such that d_0 becomes consistent w.r.t. the transformation t and d'_0 becomes consistent w.r.t. the transformation t' and we have the following pushouts for $i = 1, \dots, n$.

$$\begin{tikzcd}
B_1 \arrow[r, "j_i"] \arrow[d, "w_1"] & C_i \arrow[d, "c_i"] \\
Con_1 \arrow[r] & C'_i
\end{tikzcd}
\qquad
\begin{tikzcd}
B_2 \arrow[r, "g_i"] \arrow[d, "w_2"] & C'_i \arrow[d, "c'_i"] \\
Con_2 \arrow[r] & C''_i
\end{tikzcd}$$

So the diagrams (2) and (3) are extension diagrams. $\square$

3.5 A Component Concept for High-Level Replacement Categories

In this section the abstract framework presented in chapter 2 is applied to HLR systems. This is achieved by instantiating the generic transformation concept with HLR transformations. This also implies that the three parts of a component have to be objects of a corresponding HLR category. The consistency of the abstract framework is mapped to the definition of consistent morphisms with respect to a given transformation. The embedding diagrams of the abstract framework are mapped to the HLR extension diagrams defined in this chapter.

At this stage of the development of the component concept there are no constraints on the HLR transformations to preserve certain properties of the objects along these transformations.

The generic transformation framework repeated in section 2.2 required some properties for its instantiation. Two of them are that the concrete transformation framework $\mathcal{T}$ is closed under composition and that it includes identic transformations. Both properties are valid in the case of HLR systems, because:

1. HLR transformations are defined as sequences of direct HLR transformations and

2. applying an empty rule delivers an identic transformation.

The other four properties concern the extension property. The first one is the existence of an extension for a given transformation and a consistent morphism while the properties two to four describe the horizontal and vertical compositionality and the horizontal and vertical decomposition of extension diagrams. See section 2.2 for details. As shown in this chapter these properties hold in the case of HLR transformations if the underlying category and the class of morphisms fulfill the assumptions one to ten.

According to the instantiation of the generic transformation concept we can now formulate a more concrete notion of a specification component.

Definition 3.5.1 (HLR Component) *An* HLR component specification*, short HLR component, $COMP-HLR$ for an HLR category* $\mathbf{CAT}$ *with class of morphisms M is defined as*

$$COMP-HLR = (IMP, EXP, BOD, imp, exp),$$

where we assume that

- *IMP, EXP, BOD called* import interface, export interface, body *are objects in the category* $\mathbf{CAT}$*,*

- *exp is an HLR transformation from EXP to BOD,*

- *imp is a morphism in class M.*

Since we have the extension property for HLR transformations (see theorem 3.2.3) we can now define the composition of HLR components.

Definition 3.5.2 (HLR Composition) *Given a category* $\mathbf{CAT}$ *and a class* $M \subseteq \mathbf{CAT_{mor}}$ *such that the assumptions 1 and 2 hold and given* *two HLR components for the category* $\mathbf{CAT}$, $HLR-COMP_1$ *and* $HLR-COMP_2$, *and given a* $\mathbf{CAT}$*-transformation* $connect : IMP_1 \Rightarrow EXP_2$ *such that* imp_1 *is consistent with respect to* $connect; exp_2$, *where*

$$HLR-COMP_i = (IMP_i, EXP_i, BOD_i, imp_i, exp_i)$$

for $i \in \{1,2\}$. *Let* $xconnect = exp_2 \circ connect$. *Since* imp_1 *is consistent with respect to xconnect, we have* $xconnect' : BOD_1 \Rightarrow BOD_3$, $imp_1' : BOD_2 \rightarrow BOD_3$ *as extension of xconnect. Then we define the composition* $COMP_3 = COMP_1 \circ_{connect} COMP_3$ *as*

$$HLR-COMP_3 = (IMP_3, EXP_3, BOD_3, imp_3, exp_3),$$

where $imp_3 = imp_1' \circ imp_2$ *and* $exp_3 = xconnect' \circ exp_1$.

Remark: Since the definition of the HLR composition is based on HLR extension diagrams which are based on pushouts the result of the HLR composition is unique up to isomorphism.
The semantics of an HLR component are now defined according to section 2.3. On this way we also obtain the compositionality of semantics for the composition of HLR components.

Theorem 3.5.1 (Compositionality of Transformation Semantics) *Given a category* $\mathbf{CAT}$ *and a class* $M \subseteq \mathbf{CAT_{mor}}$ *such that the assumptions one to ten hold. If we now have a composition of HLR components for the category* $\mathbf{CAT}$ $HLR-COMP_3 = HLR-COMP_1 \circ_{connect} HLR-COMP_2$ *then we have*

$$\begin{aligned}&TrafoSem(HLR-COMP_3) =\\&TrafoSem(HLR-COMP_1) \circ Trafo(connect) \circ\\&TrafoSem(HLR-COMP_2),\end{aligned}$$

where $Trafo(connect)$ *is defined as* $Trafo(connect)(tra) = tra \circ connect$.

Proof: This follows directly from the theorems 2.5.1, 3.2.3 , 3.3.3, 3.3.4, 3.4.1 and 3.4.3. $\square$

Now the resulting component concept for HLR categories delivers a definition for the semantics of the components that is consistent with respect to composition of HLR components.

Theorem 3.5.2 (Associativity of HLR Composition) *Given a category* $\mathbf{CAT}$ *and a class* $M \subseteq \mathbf{CAT_{mor}}$ *such that the assumptions one to* *ten hold and given the HLR components for the category* **CAT**

$$HLR - COMP_i = (IMP_i, EXP_i, BOD, imp_i, exp_i)$$

for $i = 1, 2, 3$ *and* **CAT**-*transformations* $trafo_i : IMP_i \Rightarrow EXP_{i+1}$ *for* $i = 1, 2$ *then we have*

$$\begin{aligned}&(HLR - COMP_1 \circ_{trafo_1} HLR - COMP_2) \circ_{trafo_2} HLR - COMP_3 = \\ &HLR - COMP_1 \circ_{trafo_1} (HLR - COMP_2 \circ_{trafo_2} HLR - COMP_3).\end{aligned}$$

Proof: This follows directly from the theorems 2.6.1, 3.2.3 , 3.3.3, 3.3.4, 3.4.1 and 3.4.3. $\Box$

On this way the presented framework allows to model systems with suiting categorical specification techniques in a structured way. In this context the theorem above implies that the resulting system is the same if compose a set of components from the bottom to the top or the or the other way around.

Chapter 4

Instantiation to Petri Nets

This chapter covers the instantiation of the component concept for high-level replacement categories to petri nets. This is mainly achieved by showing that the concrete instantiation category satisfies the assumptions formulated in sect. 3. After these proofs the results are demonstrated with an example.

4.1 Introduction to Petri Nets

Since there are several formulations of petri nets even for the low-level case it is important to point out that a classical place/transition approach is used in this context ([25]). The main idea of this important formalism for specifying concurrent and distributed systems is the following: a net consists of places and transitions that are related by arcs. A state of the net is given by a marking of the nets places with so called token. These states can be changed by firing the nets transitions. Firing a transition means to remove tokens on the ingoing arcs of the transition and adding tokens on the outgoing arcs of the transition.

Definition 4.1.1 (Petri Nets) *A* place/transition net

$$N = (P, T, pre, post)$$

is defined by a set P of places and a set T of transitions and the functions $pre, post : T \to P^{\oplus}$, where $P^{\oplus}$ designates the free commutative monoid over P.

Definition 4.1.2 (Petri Net Morphisms) *A* place/transition net morphism $f : N_1 \to N_2$ *consists of:*

- *a function $f_P : P_1 \to P_2$ and*

- *a function* $f_T : T_1 \to T_2$,

where $N_i = (P_i, T_i, pre_i, post_i)$ *is assumed for* $i \in \{1, 2\}$ *and the diagram below has to commute for the two pre functions and the two post functions.*

$$\begin{array}{ccc} T_1 & \overset{pre_1}{\underset{post_1}{\rightrightarrows}} & P_1^{\oplus} \\ {\scriptstyle f_T}\downarrow & & \downarrow{\scriptstyle f_P^{\oplus}} \\ T_2 & \overset{pre_2}{\underset{post_2}{\rightrightarrows}} & P_2^{\oplus} \end{array}$$

Note that $f_P^{\oplus}$ *denotates the unique homomorphic extension of* f_P.

Remark: The nets and the morphisms defined above now make up the category $\mathbf{P/T}$ of place transition nets. According to definition 3.1.2 a rule for petri net transformations now consists of three place/transition nets and two morphisms $L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$. A place/transition net transformation is then given by a double pushout diagram according to definition 3.1.3 in the category $\mathbf{P/T}$.

4.2 Instantiation

Before discussing the results of this instantiation it is important to show that the category of low level petri nets, the category $\mathbf{P/T}$, together with the class M_{inj} which contains all injective low level petri net morphisms fulfills all assumptions formulated in the previous chapters. By the definition of M_{inj} we already obtain that assumptions 4, 3 and 6 are valid.

Theorem 4.2.1 (Petri Net Pushouts) *Given three petri nets* A, B *and* C *and the petri net morphisms* $A \leftarrow B \rightarrow C$ *then there is a pushout* $A \rightarrow D \leftarrow C$.

Remark: Note that this theorem is even stronger than assumption 1. The proof is omitted here. The idea is to construct the pushout componentwisely for the sets of places and the sets of transformations. A proof for the pushout property of this construction can be found in [20].

Theorem 4.2.2 (Inheritance of Monomorphisms under PO's) *For each pushout diagram in* $\mathbf{P/T}$ *as below,* $B \to C \in M_{inj}$ *implies* $A \to D \in$

M_{inj}.

$$\begin{tikzcd} B \arrow[r] \arrow[d] & C \arrow[d] \\ A \arrow[r] & D \end{tikzcd}$$

Remark: This theorem corresponds to assumption 2. It is valid because we have this property for the category **SET** and petri nets are based on sets. See [20] for details.

Theorem 4.2.3 (Existence of Initial Pushouts I) *Given two nets N_1 and N_2 with a petri net inclusion $d : N_1 \to N_2$. Then we get the nets B and C with morphisms i, i', d' such that the diagram below is an initial pushout with respect to d.*

$$\begin{tikzcd} B \arrow[r, "i"] \arrow[d, "d'"] & N_1 \arrow[d, "d"] \\ C \arrow[r, "i'"] & N_2 \end{tikzcd}$$

Proof: Given an inclusion $d : N_1 \to N_2$. Then let the net B defined as:

$$\begin{aligned} B_T &= \emptyset \\ B_P &= \{p \in N_{1P} | \exists t \in N_{2T} \bullet (d(p) \in pre(t) \cup post(t)) \wedge (\neg \exists t' \in \\ &\quad N_{1T} \bullet d(t') = t)\} \\ pre_B &= \emptyset \\ post_B &= \emptyset \end{aligned}$$

Note that pre_B and $post_B$ are well defined since $B_T = \emptyset$. Moreover, a net C has to be defined:

$$\begin{aligned} C_T &= \{t' \in N_{2T} | \neg \exists t \in N_{1T} \bullet d(t) = t'\} \\ C_P &= B_P \cup \{p' \in N_{2P} | \neg \exists p \in N_{1P} \bullet d(p) = p'\} \\ pre_C(t) &= pre_{N_2}(t) \\ post_C(t) &= post_{N_2}(t) \end{aligned}$$

By the construction of the net C we know that $C \cup N_1 = N_2$. Given an arbitrary transformation $t \in C_T$ and a place $p \in pre_C(t) \cup post_C(t)$. Then we have that $p \in N_2$. So the following case distinction is complete.

1. $p \in C_P$ – in this case pre_C and $post_C$ respectively would be well defined.

2. $p \in N_1$ – By the definition of C we can conclude $t \notin N_1$. Then the definition of B implies $p \in B_P$ and so we have $p \in C_P$ what corresponds to the first case of the case distinction.

Now we can conclude that pre_C and $post_C$ are well defined. It is easy to see that the net B is fully included in the net N_1 and the net C. So we can define inclusions $i : B \to N_1$ and $d' : B \to C$. This implies that $i \in M_{inj}$. Moreover, the net C is a subnet of the net N_2, so we obtain an inclusion $i' : C \to N_2$. This leads to the diagram (I) below.

$$\begin{tikzcd}
B \arrow[r, "i"] \arrow[d, "d'"] \arrow[dr, phantom, "(I)"] & N_1 \arrow[d, "d"] \arrow[ddr, bend left, "k_2"] & \\
C \arrow[r, "i'"] \arrow[drr, bend right, "k_1"] & N_2 \arrow[dr, "k"] & \\
& & K
\end{tikzcd}$$

Next it is shown that the constructed diagram (I) is a pushout. Since all morphisms are petri net inclusions the diagram commutes. Assumed there is an object K with morphisms k_1 and k_2 as above such that $k_1 \circ d' = k_2 \circ i$. Now we define a morphism $k : N_2 \to K$ as follows.

$$k(n) = \begin{cases} \underline{if} & n \in C \quad \underline{then} \quad k_1(c) \\ \underline{else} & k_2(n) \end{cases}$$

Now given an arbitrary $c \in C$. Then we have:

$$k \circ i'(c) = k(i'(c)) = k(c) = k_1(c).$$

So we have shown that $k_1 = k \circ i'$. Next it is shown that $k_2 = k \circ d$. Given an arbitrary $n \in N_1$. Then we have:

$$k \circ d(n) = k(d(n)) = k(n).$$

Since B is a subnet of N_1 the following case distinction is complete

1. $n \notin B$ – By the definition of C we can conclude that $n \notin C$. So we have $k(n) = k_2(n)$.

2. $n \in B$ – In this case we know that $k_1(d(n)) = k_2(i(n))$. Since i and d are inclusions we have that $k_1(n) = k_2(n)$. Together with the definition of k we can conclude now $k(n) = k_1(n) = k_2(n)$

We have just shown that $k_2 = k \circ d$ holds. It remains to show that k is unique under the shown properties. Assumed there is a $k' : N_2 \to K$ such that $k_1 = k' \circ i'$ and $k_2 = k' \circ d$. The assumption on k' imply that $k' \circ i' = k \circ i'$ and $k' \circ d = k \circ d$. So we have:

$$\begin{array}{lll} & \forall c \in C \bullet k'(i'(c)) = k(i'(c)) & \\ \Rightarrow & \forall c \in C \bullet k'(c) = k(c) & (i' \text{ is inclusion}) \end{array}$$

$$\begin{array}{lll} & \forall c \in N_1 \bullet k'(d(c)) = k(d(c)) & \\ \Rightarrow & \forall c \in N_1 \bullet k'(c) = k(c) & (d \text{ is inclusion}) \end{array}$$

Since $N_2 = C \cup N_1$ we have that $k = k'$ what implies that k is unique under the given properties. So the diagram (I) is a pushout. Given now another pushout (1) over the inclusion d as depicted below, where j_1 is assumed to be a monomorphism.

$$\begin{tikzcd}
B \arrow[rr, "i"] \arrow[dr, "\overline{b}"] \arrow[ddd, "d'"'] & & N_1 \arrow[ddd, "d"] \\
& B' \arrow[ur, "j_1"'] \arrow[d, "j_2"'] \arrow[l, phantom, "(2)"', shift right=4] \arrow[r, phantom, "(1)", shift right=4] & \\
& C' \arrow[dr, "j_3"] & \\
C \arrow[ur, "\overline{c}"] \arrow[rr, "i'"'] & & N_2
\end{tikzcd}$$

It is known that j_1 and d are injective per assumption. Moreover, $j_3 \circ j_2 = d \circ j_1$ must hold since (1) is a pushout. This implies that $j_3 \circ j_2$ is injective as well. Then we can conclude that j_2 is injective. Since (1) is a pushout and we have the theorem 4.2.2 we know that j_3 is an injective petri net morphism.

Lemma 4.2.1 $\forall b \in B \ \exists b' \in B' \bullet j_1(b') = b$

Proof: Since B_T is empty b is a place of B_P.

- Given $b \in B_P$.
- By the construction of B it is known that if $b \in B_P$ then $\exists t \in N_2 \bullet b \in pre(t) \cup post(t)$ and $t \notin N_1$.

- By the pushout property of (1) we can conclude that $\exists t' \in C' \bullet j_3(t') = t \in N_2$.
- Since j_3 is a petri net morphism the following property must hold: $\exists \overline{b} \in C'$ such that $\overline{b} \in pre_{C'}(t')$, and $\overline{b} \in post_{C'}(t')$ respectively and $j_3(\overline{b}) = b \in N_2$.
- Now we have $b \in B \subseteq N_1 \subseteq N_2$ and $\overline{b} \in C'$ with $j_3(\overline{b}) = b$. That means $b \in N_1$ and $\overline{b} \in C'$ have a common image in N_2. Since (1) is a pushout of injective morphisms we obtain $\exists b' \in B' \bullet j_1(b') = b$, following the pushout characterizations section A.1, taken from [7]. $\surd$

Lemma 4.2.2 $\forall c \in C\ \exists c' \in C' \bullet j_3(c') = c$

Proof:

- Given $c \in C$. Note that c can be a place or a transition. By the construction of C it is known that the following case distinction is complete.
- First case: $c \in B$. In the proof above a place $\overline{b} \in C'$ was constructed. Choosing $b = c$ then delivers $j_3(\overline{b}) = b$. So we have a $c' \in C'$ with $j_3(c') = c$. $\surd$
- Second case: $c \in N_2$ and $c \notin N_1$. Since (1) is a pushout we obtain $\exists c' \in C' \bullet j_3(c') = c$. $\surd$

Knowing that j_1 and j_3 are injective we can define the following partial petri net morphisms $j_1^{-1} : N_1 \nrightarrow B'$ and $j_3^{-1} : N_2 \nrightarrow C'$.

$$j_1^{-1}(n) = \underline{if}\ \exists n' \in B' \bullet j_1(n') = n\ \underline{then}\ n'$$
$$j_3^{-1}(n) = \underline{if}\ \exists n' \in C' \bullet j_3(n') = n\ \underline{then}\ n'$$

By the two Lemmas above we can conclude that j_1^{-1} is defined for all $b \in B \subseteq N_1$ and that j_3^{-1} is defined for all $c \in C \subseteq N_2$. Moreover we have $j_1 \circ j_1^{-1} = id_{N_1}$ and $j_3 \circ j_3^{-1} = id_{N_2}$ and we know that j_1^{-1}, j_3^{-1} are injective. So we can compose the following petri net morphisms.

1. $\overline{b} : B \to B'$ with $\overline{b} = j_1^{-1} \circ i$ is well defined and total since j_1^{-1} is defined for B and so for $i(B)$. Moreover, this morphism is injective since i and j_1^{-1} are injective. $\Rightarrow \overline{b} \in M_{inj}$
2. $\overline{c} : C \to C'$ with $\overline{c} = j_3^{-1} \circ i'$ is well defined and total since j_1^{-1} is defined for C and so for $i'(C)$. Moreover, this morphism is injective since i' and j_3^{-1} are injective. $\Rightarrow \overline{c} \in M_{inj}$

By defining the morphisms in that way we also obtain that $i = j_1 \circ \overline{b}$ and $i' = j_3 \circ \overline{c}$ because:

$$\begin{array}{lll} & \overline{b} = j_1^{-1} \circ i & \text{(definition } \overline{b}\text{)} \\ \Rightarrow & j_1 \circ \overline{b} = j_1 \circ j_1^{-1} \circ i & \\ \Rightarrow & j_1 \circ \overline{b} = i & \text{(definition } j_1^{-1}\text{)} \end{array}$$

$$\begin{array}{lll} & \overline{c} = j_3^{-1} \circ i' & \text{(definition } \overline{c}\text{)} \\ \Rightarrow & j_3 \circ \overline{c} = j_3 \circ j_3^{-1} \circ i' & \\ \Rightarrow & j_3 \circ \overline{c} = i' & \text{(definition } j_3^{-1}\text{)} \end{array}$$

Now it remains to show that the diagram (2) as depicted above is a pushout. By the construction of the large square it is known that $d \circ i = i' \circ d'$

$$\begin{array}{lll} \Rightarrow d \circ j_1 \circ \overline{b} & = j_3 \circ \overline{c} \circ d' & \text{Definition of } \overline{b}, \overline{c} \\ \Rightarrow j_3 \circ j_2 \circ \overline{b} & = j_3 \circ \overline{c} \circ d' & \text{(1) is a pushout} \\ \Rightarrow j_2 \circ \overline{b} & = \overline{c} \circ d' & j_3 \text{ is a monomorphism} \end{array}$$

Now it is known that (2) is a commutative diagram of monomorphisms. Following the ideas of [7] it is now sufficient to show two things for proving the pushout property of (2).

1. According to section A.1 a diagram that fulfills the following condition is called *epimorphic*. The condition adapted to the concrete case is $\overline{c}(C) \cup j_2(B') = C'$, where $\overline{c}(C)$ designates the union of all images of $\overline{c} : C \to C'$ in C' and $j_2(B')$ designates a corresponding set for j_2. This condition is the same as

   $$x \in \overline{c}(C) \vee x \in j_2(B) \Leftrightarrow x \in C'.$$

   The arrow from the left to the right is valid because both morphisms only have images in C'. The other direction is shown by the following case distinction that is based on $C \subseteq N_2$.

   (a) Let $j_3(x) \in C$.

   $$\begin{array}{ll} \Rightarrow \exists c \in C \bullet i'(c) = j_3(x) & i' \text{ is inclusion} \\ \Rightarrow j_3 \circ \overline{c}(c) \ = \ j_3(x) & \text{construction of } \overline{c} \\ \Rightarrow \overline{c}(c) \ = \ x & j_3 \text{ is monomorphism} \\ \Rightarrow x \in \overline{c}(C) \Rightarrow x \in \overline{c}(C) \vee x \in j_2(B') & \end{array}$$

(b) Let $j_3(x) \notin C$.

$\Rightarrow j_3(x) \in N_1$	the large pushout square
$\Rightarrow \exists n \in N_1 \bullet d(n) = j_3(x)$	d is inclusion
$\Rightarrow \exists b' \in B' \bullet j_2(b') = x \wedge j_1(b') = n$	(1) is pushout of injective morphisms
$\Rightarrow x \in j_2(B') \Rightarrow x \in \overline{c}(C) \vee x \in j_2(B')$	

2. According to section A.1 the next condition on the diagram is called *reduced chain condition.* For all $c \in C, b' \in B'$ with $\overline{c}(c) = j_2(b')$ there is a $b \in B$ such that $d'(b) = c \wedge \overline{b}(b) = b'$. Given $c \in C, b' \in B'$ with $\overline{c}(c) = j_2(b')$ two cases can occur.

   (a) Let $c \in N_1$.
   By the construction of B we can conclude that $c \in B$, i.e. $d'(c) = c$.

$\overline{c} \circ d'(c) = j_2 \circ \overline{b}(c)$	commutativity of (2)
$\Rightarrow \overline{c}(c) = j_2 \circ \overline{b}(c)$	d' is inclusion
$\Rightarrow j_2(b') = \overline{c}(c) = j_2 \circ \overline{b}(c)$	assumption for reduced chain condition proof
$\Rightarrow j_2(b') = j_2 \circ \overline{b}(c)$	
$\Rightarrow b' = \overline{b}(c)$	j_2 is injective
$\Rightarrow d'(c) = c \wedge \overline{b}(c) = b'$	

   By selecting $b = c$ we have a b that fulfills the needed property.

   (b) Let $c \notin N_1$. The construction of B now implies $c \notin B$. Let $\overline{c}(c) = c' = j_2(b')$. We have:

$i'(c) = j_3(\overline{c}(c))$	construction of $\overline{c}$
$i'(c) = c$	construction of $\overline{i'}$
$\Rightarrow j_3(c') = c$	(*)

   The pushout property of (1) implies $j_3(j_2(b')) = d(j_1(b'))$

$\Rightarrow j_3(c') = d(j_1(b'))$	definition c'
$\Rightarrow c = j_1(b')$	d is inclusion and (*)
$\Rightarrow c \in N_1$	

   This is a contradiction to the assumption made above. As a consequence the reduced chain condition is valid in this case.

Since the reduced chain condition is proved for the diagram (1), this diagram is a pushout. Now given two morphisms $a' : B \to B'$ and $b' : B \to B'$, such that:

1. $i = j_1 \circ a'$ and

2. $i' = j_3 \circ b'$ and

3. $B' \xrightarrow{j_3} C' \xleftarrow{b'} C$ is a pushout over $B' \xleftarrow{a'} B \xrightarrow{d'} C$.

Since we know that $i = j_1 \circ \overline{b}$ and $i' = j_3 \circ \overline{c}$ we have:

$$j_1 \circ b' = j_1 \circ \overline{b}$$
$$j_3 \circ c' = j_3 \circ \overline{c}$$

Now the monomorphism property of j_1 and j_3 implies $b' = \overline{b}$ and $c' = \overline{c}$. So we have the uniqueness of $\overline{b}$ and $\overline{c}$ under the corresponding properties. This completes the proof for the initiality of the pushout (I) constructed on page 49. $\square$

Theorem 4.2.4 (Existence of Initial Pushouts II) *Given two nets N_1 and N_2 with an injective petri net morphism $d : N_1 \to N_2$. Then we get the nets B and C with morphisms i, i', d' such that the diagram below is an initial pushout with respect to d.*

$$\begin{tikzcd} B \arrow[r, "i"] \arrow[d, "d'"] & N_1 \arrow[d, "d"] \\ C \arrow[r, "i'"] & N_2 \end{tikzcd}$$

Proof: Since we have a corresponding theorem for inclusions (4.2.3) we can conclude by section A.2 that the theorem is valid for monomorphisms as well. $\square$

Remark: On this way the category **P/T** together with the class M_{inj} fulfills assumption 5.

Theorem 4.2.5 (Petri Net Pullbacks) *Given the nets B, C, D with injective petri net morphisms $b : B \to D, c : C \to D$. Now we have a net A with morphisms $a_1 : A \to C$ and $a_2 : A \to B$ such that the diagram below becomes a pullback diagram.*

$$\begin{tikzcd} A \arrow[r, "a_2"] \arrow[d, "a_1"] \arrow[dr, phantom, "(P)"] & B \arrow[d, "b"] \\ C \arrow[r, "c"] & D \end{tikzcd}$$

Theorem 4.2.6 (Inheritance of Monomorphisms under PB's) *If we have a pullback diagram as (P) above then the following implication holds. If c and b are in class M_{inj} then also a_1 and a_2 are in class M_{inj}.*

Theorem 4.2.7 (Pushouts are Pullbacks) *If in the diagram (P) above (D, c, b) is a pushout of (A, a_1, a_2) and a_1, a_2 are injective petri net morphisms then (A, a_1, a_2) is a pullback of (D, c, b).*

Theorem 4.2.8 (Pushout Pullback Decomposition) *Given a pushout (1)+(2) as depicted below, where (2) is assumed to be a pullback and the morphisms a_1, b_1, e, b_2, d are assumed to be injective. Now we have that (1) is a pushout.*

$$\begin{tikzcd} A \arrow[r, "a_2"] \arrow[d, "a_1"] \arrow[dr, phantom, "(1)"] & B \arrow[r, "b_2"] \arrow[d, "b_1"] \arrow[dr, phantom, "(2)"] & E \arrow[d, "e"] \\ C \arrow[r, "c"] & D \arrow[r, "d"] & F \end{tikzcd}$$

Remark: These theorems correspond to the assumptions 8, 7, 9 and 10. Their validity follows from the the fact that the category **P/T** is based on sets and the corresponding properties hold in **SET**. See [6] for detailed proofs.

Now we have the following results. A petri net component now consists of three place/transition nets:

- a net IMP, called import interface,
- a net EXP, called export interface and
- a net BOD, called body.

Moreover, the component contains an injective petri net morphism $imp : IMP \to BOD$ and a **P/T** transformation $exp : EXP \Rightarrow BOD$.

The semantics are then given by the extension of all possible import transformations. Since the category **P/T** is a valid instantiation of the HLR transformation based component concept we have the compositionality of the semantics according to section 2.5. This concept is very similar to the independently developed one in [21] which is based on a special notion of morphisms. These morphisms then realize a refinement of the export interface in the body. This corresponds to the idea of the transformation based refinement used here.

4.3 Example

This section presents parts of a component based specification for the IOSIP[1] case study which has its domain in production automation. In fact, this case study on holonic production systems has been proposed in [1] as a reference case study in the national priority program on "Integration of Software Specifications for Applications in Engineering" sponsored by the German research council ([5]). The following solution for this case study applies components of place/transition nets. This means that our components consist of import, export and body nets and corresponding connections that are based on Petri net morphisms and transformations.

A complete UML based model for the same case study with an informal component concept can be found in [18]. Since we have strong theoretical foundations for Petri nets the approach presented here should be regarded as an alternative model based on formal specification techniques including a formal component concept.

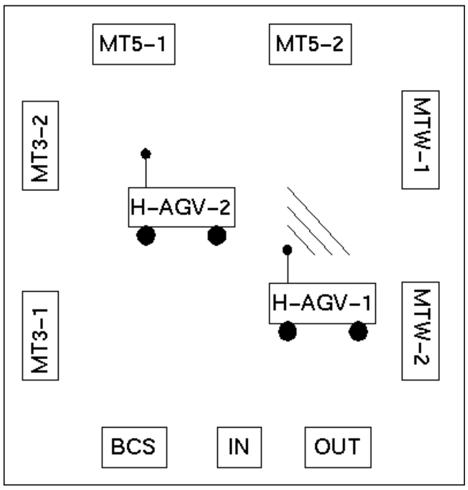

Figure 4.1: Layout of the production system

The main objective of this study was to specify a control software for so called holonic automated guided vehicles, short H-AGVs. These H-AGVs have the task to transport workpieces within a production environment that

[1]Integration of object-oriented software specification techniques and their application-specific extension for industrial production systems on the example of automobile industry

produces engines for automobiles. Figure 4.1 shows the layout of the production system in the case study, where the following domain objects occur:

- MT-3 — two machine tools (3 axle drilling machines),
- MT-5 — two machine tools (5 axle drilling machines),
- MTW — machines that wash the work pieces,
- BCS — a battery charging station for the H-AGVs,
- IN — an input stock,
- OUT — an output stock,
- H-AGV — holonic automated guided vehicles.

The workpieces have to be brought from the input stock to the three different machine types (MT-3, MT-5 and MTW) by the H-AGVs. If one of this machines has finished processing a workpiece then it moves the workpiece to its outgoing buffer and reports the new buffer assignment to the H-AGVs. At this point the H-AGVs start a negotiation via a wireless broadcast medium to decide which H-AGV is nearest to the corresponding machine and so in duty to pick up the workpiece and to bring it to the next machine. If the processing of a workpiece is finished in the production system it has to be brought to the outgoing stock. See [1] for the full description of the requirements for the case study.

Since we deal with embedded systems it is important to model the environment and the system itself in two steps. Correspondingly, the specification is divided into a domain model and a software model. The software model is based on algebraic high-level nets which are the subject of Chapter 6. that part of the example can be found in sect. 6.2.

The place/transition nets below specify parts of the domain model. In contrast to other modeling techniques components are here already used in the domain model which documents the analysis of the requirements. This is an advantage because we need only one structuring technique for both, the domain model and the software model. An overview of the needed component structure that specifies the behavior of the complete system (with one H-AGV) is given in Figure 4.2, where the arrows symbolize a composition of the corresponding components. Note that multiple ingoing and outgoing arcs for single components mean that this component has multiple export and import interfaces.

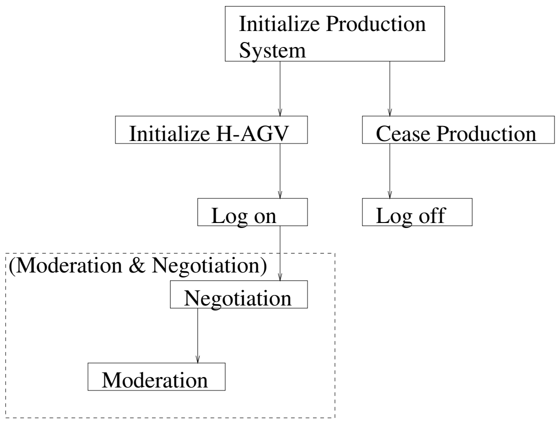

Figure 4.2: Overview of domain model components

Now we have a closer look on two sample components, namely the component *Negotiation* and the component *Moderation*. The component *Negotiation* $= (IMP_1, EXP_1, BOD_1, imp_1, exp_1)$ describes the behavior of an H-AGV in the negotiations mentioned above.

Figure 4.3 shows the export net of the *Negotiation* component. In this net we have a place *ready*. A token on this place expresses that the corresponding H-AGV is ready. If now one machine sends a work piece order the H-AGVs have to conduct a moderated discussion. For this purpose a moderator has to be selected from those H-AGVs that are ready. Since the case study requirements do not constrain how this selection takes place I just had to model the possibilities that a concrete H-AGV is selected as moderator or not. This was done by inserting corresponding transitions into the net. In the case that an H-AGV is not selected as moderator for a possible transport order it receives an offer for the corresponding order from the moderator. If the H-AGV is able to make a better offer it sends it to the moderator who decides whether the offer is accepted or not. In both cases the H-AGV is ready afterwards. This is also true for the case that the H-AGV has no better offer to make.

If a submitted offer is accepted the corresponding H-AGV has to execute the resulting transport order. This is modeled in the body of the *Negotiation* component. The body net and the export net are related by the transformation exp_1 which is based on the rule in Figure 4.4. Figure 4.6 shows the complete double pushout transformation, where EXP_1 and BOD_1 designate the export and the body of the *Negotiation* component.

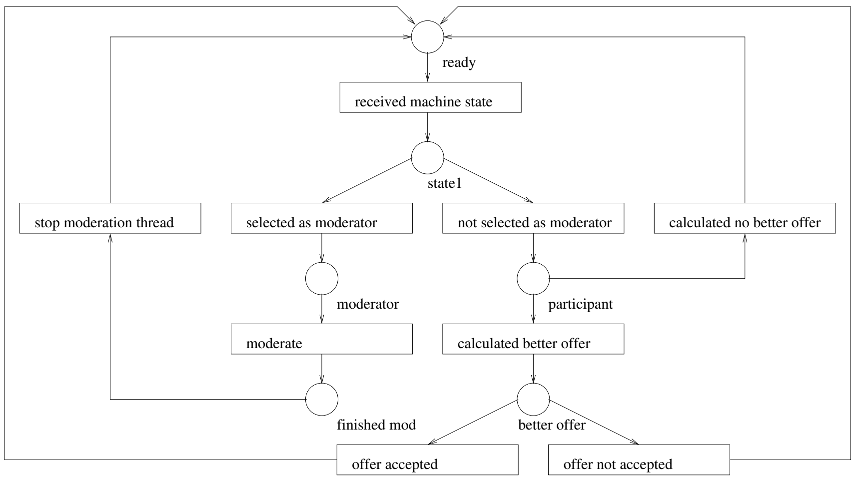

Figure 4.3: Export of component *Negotiation*

The transitions *calculated better offer* and *offer accepted* are replaced by a more detailed model that also covers the execution of transport orders and the explicit submission of offers.

The import of the component *Negotiation* is completely embedded into the body of this component. The connection of the import with the export of other components allows a further refinement of the transition *moderate*.

The component *Moderation* $= (IMP_2, EXP_2, BOD_2, imp_2, exp_2)$ now models a refinement of the transition *moderate*. Accordingly, this transition is part of the export net of the *Moderation* component and is refined along

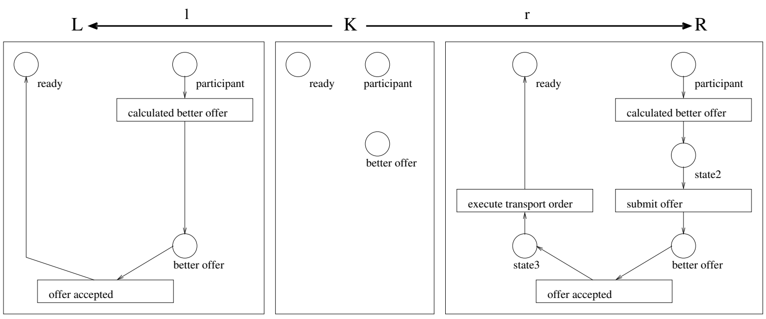

Figure 4.4: Rule for exp_1 in component *Negotiation*

the transformation exp_2 which is based on the rule in Figure 4.9. Since the left side of the rule is equal to the export of the component we have that the transformation exp_2 is equal to the rule in Figure 4.9. The rule deletes the transition *moderate* and adds a subnet that models the full moderation procedure. This net works as follows: first of all the moderator H-AGV sends an initial offer to the other H-AGVs. If now one or more better offers are received the moderator selects the best one and confirms it after a timeout. If no other offer has been submitted to the moderator within the timeout the moderator executes the transport order for himself. The import net IMP_2 of component *Moderation* is empty.

Since the net in the import of the *Negotiation* component and the export of the *Moderation* component are equal we can compose them via an identical transformation and obtain a component *Negotiation & Moderation*, where the body of this component is depicted in Figure 4.11. This net is constructed as a gluing of the body nets of the two components *Negotiation* and *Moderation* over the parts identified by the connecting transition. The export net of this composed component is given by the export of the *Negotiation* component. As import we obtain an empty net.

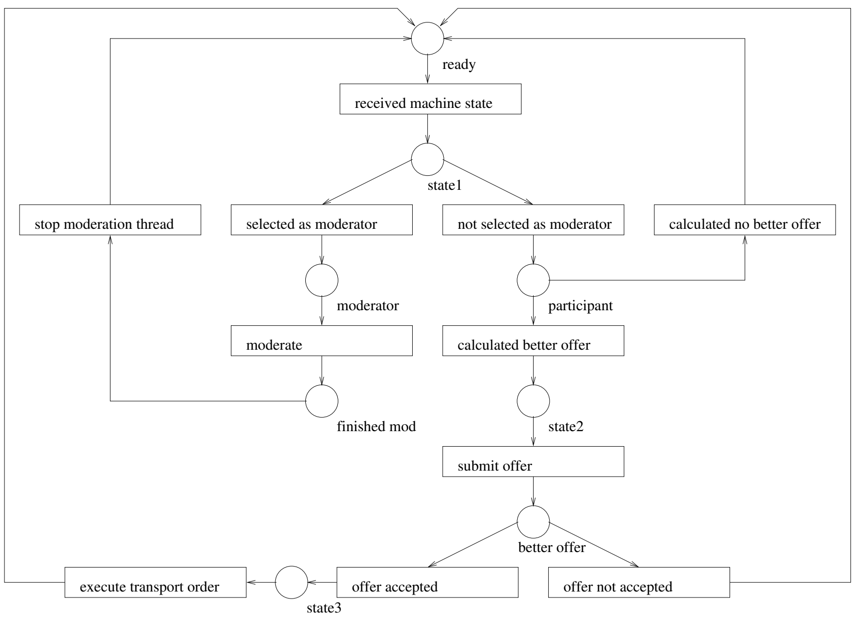

Figure 4.5: Body of component *Negotiation*

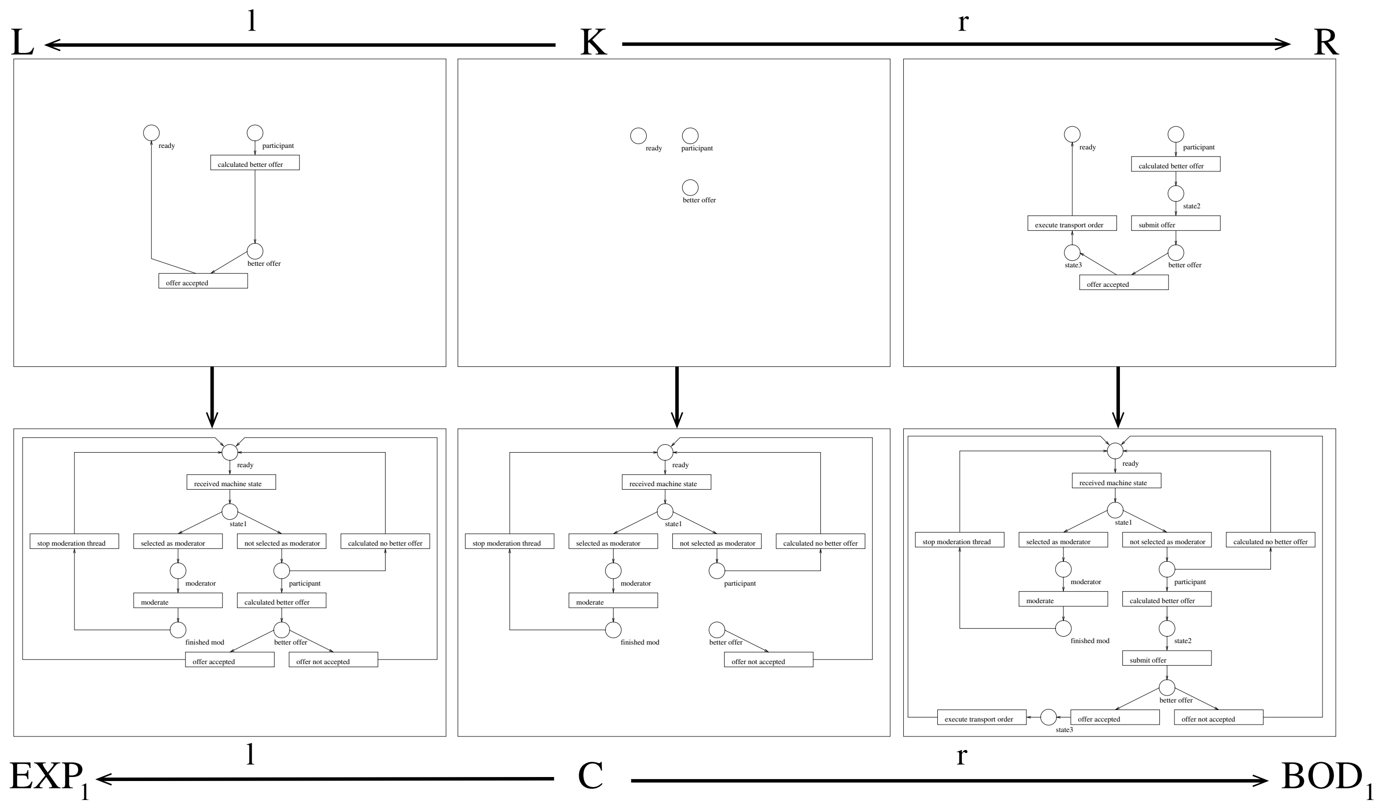

Figure 4.6: Transformation exp_1

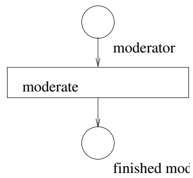

Figure 4.7: Import of component *Negotiation*

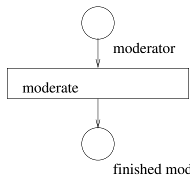

Figure 4.8: Export of component *Moderation*

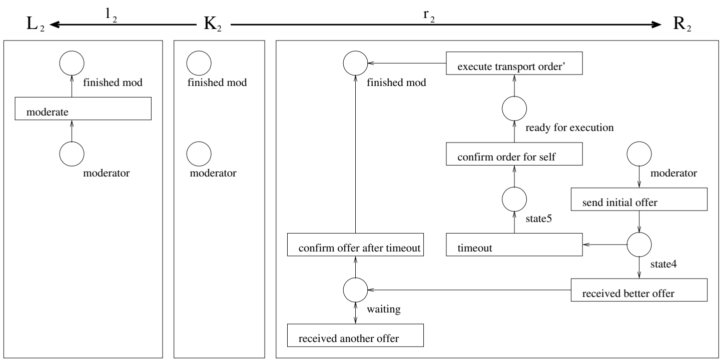

Figure 4.9: Rule for exp_2 in component *Moderation*

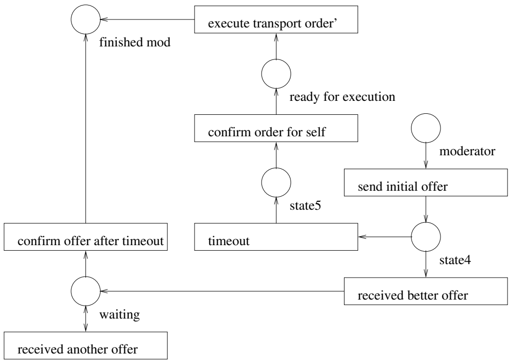

Figure 4.10: Body of component *Moderation*

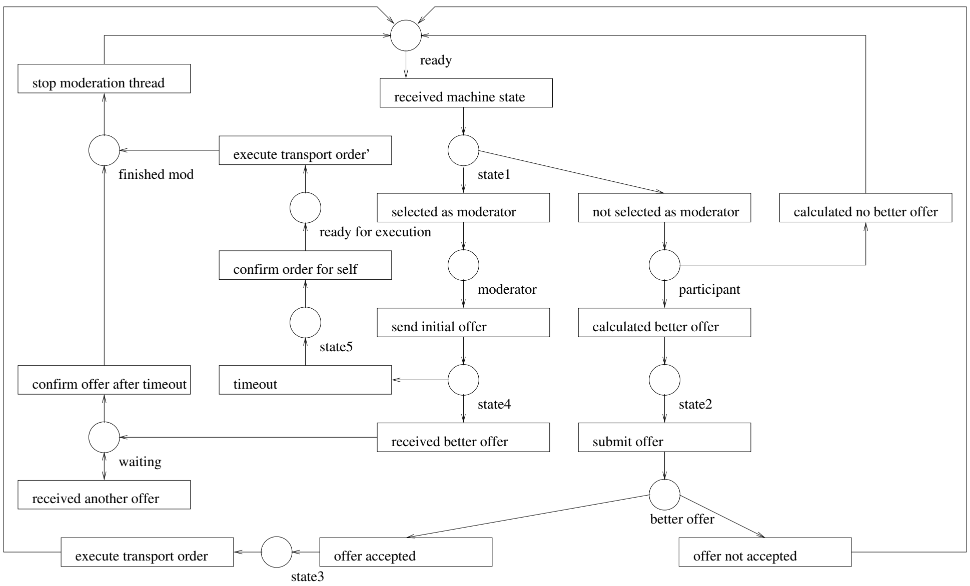

Figure 4.11: Body of component *Negotiation & Moderation*

Chapter 5

Instantiation to Algebraic Specifications

Algebraic specifications in the sense of [8] are a well known technique for specifying complex data structures. In [4] HLR Systems have already been applied to algebraic specifications. But since the presented component framework requires additional properties some more proofs have to be done. Unfortunately some assumptions (e.g. 5 and 9) do not hold for the algebraic specifications according to [8]. Due to that we use a slightly modified notion of algebraic specification morphisms, which is based on explicit mappings for the equations. This kind of definition was already introduced as a category **SPEC3** in [14].

5.1 Introduction to Algebraic Specifications

Besides the mentioned differences of the used notions we still have that specifications consist of signatures and equations. So we have to define the signatures and signature morphisms first.

Definition 5.1.1 (Signature) *A* signature $\Sigma = (S, OP)$ *consists of*

- *a set of sorts S and*
- *a set OP of constants and operator symbols over S, where each $op \in OP$ has the structure $op : w \to s$ with $w \in S^*$ and $s \in S$.*

Definition 5.1.2 (Signature Morphism) *A* signature morphism $f_\Sigma : \Sigma_1 \to \Sigma_2$ *consists of two functions* $f_S : S_1 \to S_2$ *and* $f_{OP} : OP_1 \to OP_2$

such that the following implication holds:

> **if** $\sigma : s_1 \dots s_n \to s \in OP_1$,
> **then** $f_{OP}(\sigma) : f_S(s_1) \dots f_S(s_n) \to f_S(s) \in OP_2$.

(for $\Sigma_i = (S_i, OP_i)$ *and* $i \in \{1, 2\}$*)*

Remark: The signatures and the signature morphisms make up the category **Sig**.

Since equations consist of terms we have to define terms first.

Definition 5.1.3 (Terms) *Given a signature* $\Sigma = (S, OP)$ *and a corresponding sorted set of variables* $X = \bigcup_{s \in S} X_s$. *Now the set of* terms *of sort* s *over* Σ *and* X*, called* $T_\Sigma(X)_s$*, is defined as follows:*

- $\sigma \in T_\Sigma(X)_s$ *for all* $\sigma : \to s \in OP$
- $X_s \subseteq T_\Sigma(X)_s$
- **if** $t_i \in T_\Sigma(X)_{s_i}$ *for* $i \in \{1, \dots, n\}$ *and* $\sigma : s_1 \dots s_n \to s \in OP$, **then** $\sigma(t_1, \dots, t_n) \in T_\Sigma(X)_s$

Equations are now defined by two terms of the same sort $t_1 = t_2$. It is easy to see that a signature morphism $f_\Sigma : \Sigma_1 \to \Sigma_2$ can be extended to a term morphism $f_\Sigma^\# : T_{\Sigma_1}(X) \to T_{\Sigma_2}(X)$. According to that we can now translate equations over a signature Σ_1 and variables X into equations over signature Σ_2 and variables X, $t_1 = t_2 \longmapsto f_\Sigma^\#(t_1) = f_\Sigma^\#(t_2)$. On this way we obtain morphisms of equations.

Definition 5.1.4 (Specification) *An* algebraic specification $SPEC = (\Sigma, E)$ *consists of a signature* Σ *and a set of equations* E *over* Σ.

Definition 5.1.5 (Specification morphism) *A* specification morphism $f : SPEC_1 \to SPEC_2$*, where* $SPEC_i = (\Sigma_i, E_i)$ *for* $i \in \{1, 2\}$*, is given by a signature morphism* $f_\Sigma : \Sigma_1 \to \Sigma_2$ *and an equation morphism* $f_E : E_1 \to E_2$ *such that* $f_E(t_1 = t_2) = (f_\Sigma^\#(t_1) = f_\Sigma^\#(t_2))$.

Remark: At this point we have the concrete difference between the notions in [8, 9] and the one used here, because in the more classical definitions we do not have equation morphisms as part of the specification morphisms.

Note that algebraic specifications and specification morphisms make up the category **Spec**. According to definition 3.1.2 a rule for specification transformations now consists of three specifications and two morphisms $L \xleftarrow{l} K \xrightarrow{r} R$. A specification transformation is then given by a double pushout diagram according to definition 3.1.3 in the category **Spec**.

5.2 Instantiation

In the following the assumptions made in the previous chapters are proven for the category **Spec** together with the class M_{inj} that contains all injective specification morphisms. By that we already fulfill assumptions 4, 3 and6.

Theorem 5.2.1 (Algebraic Specification Pushouts) *Given three specifications A, B and C and the specification morphisms $A \leftarrow B \rightarrow C$ then there is a pushout $A \rightarrow D \leftarrow C$.*

Remark: This theorem corresponds to assumption 1. Again the idea is to construct the pushout componentwisely for the underlying sets.

Theorem 5.2.2 (Inheritance of Monomorphisms under PO's) *For each pushout diagram in* **Spec** *as below, $B \rightarrow C \in M_{inj}$ implies $A \rightarrow D \in M_{inj}$.*

$$\begin{tikzcd} B \arrow[r] \arrow[d] & C \arrow[d] \\ A \arrow[r] & D \end{tikzcd}$$

Remark: This theorem corresponds to assumption 2. It's validity can be concluded from the category **SET**, because **Spec** is based on sets.

Theorem 5.2.3 (Existence of Initial Pushouts I) *Given two specifications $SPEC_1$ and $SPEC_2$ with an inclusion $d : SPEC_1 \rightarrow SPEC_2$. Then we get the specifications B and C with morphisms i, i', d' such that the diagram below is an initial pushout with respect to d.*

$$\begin{tikzcd} B \arrow[r, "i"] \arrow[d, "d'"] & N_1 \arrow[d, "d"] \\ C \arrow[r, "i'"] & N_2 \end{tikzcd}$$

Proof: Given two specifications $SPEC_1$ and $SPEC_2$, such that $SPEC_i = (S_i, OP_i, E_i)$ for $i \in \{1,2\}$, and an inclusion $d : SPEC_1 \to SPEC_2$. Then we construct the specification $B = (S_B, OP_B, E_B)$ as follows.

$$\begin{array}{lcl} S_B & = & \{s \in S_1 | \exists op \in OP_2 \bullet op : s_1 \dots s_n \to s' \wedge s \in \\ & & \{s_1, \dots, s_n, s'\} \wedge op \notin OP_1\} \cup \\ & & \{s \in S_1 | \exists op \in OP_1 \bullet op : s_1 \dots s_n \to s' \wedge s \in \\ & & \{s_1, \dots, s_n, s'\} \wedge \exists e \in E_2 \bullet e \notin E_1 \wedge op \in ops(e)\} \\ OP_B & = & \{op \in OP_1 | \exists e \in E_2 \bullet e \notin E_1 \wedge op \in ops(e)\} \\ E_B & = & \emptyset \end{array}$$

This definition used the mapping $ops : E \to \mathcal{P}(OP)$ that delivers the set of all the operation symbols used in the terms of a given equation. S_B now contains all sorts of S_1 for that a new sort was added along the inclusion d or that are used in the type of an operation symbol for that a new equation was added along the inclusion d. These operation symbols then define the set of operation symbols in the specification B. This specification is now the boundary of the inclusion $d : SPEC_1 \to SPEC_2$. Next, the context of this inclusion, the specification $C = (S_C, OP_C, E_C)$ is constructed.

$$\begin{array}{lcl} S_C & = & S_B \cup \{s \in S_2 | s \notin S_1\} \\ OP_C & = & OP_B \cup \{op \in OP_2 | op \notin OP_1\} \\ E_C & = & \{e \in E_2 | e \notin E_1\} \end{array}$$

Now we can define the inclusions $d' : B \to C$, $i : B \to SPEC_1$ and $i' : C \to SPEC_2$, as depicted below. $\Rightarrow i \in M_{inj}$

$$\begin{tikzcd} B \arrow[r, "i"] \arrow[d, "d'"] & SPEC_1 \arrow[d, "d"] \\ C \arrow[r, "i'"] & SPEC_2 \end{tikzcd}$$

Next we construct the pushout $K = (S_K, OP_K, E_K)$ over $C \xleftarrow{d'} B \xrightarrow{i} SPEC_1$. This is done in two steps. First we construct the pushout for the signatures according to [8]. In the second step we construct the pushout for the three equation sets. In this context we regard the inclusion i as the needed inclusion for the construction of pushouts in the category **Sig** following the ideas of the *parameter passing* ([8]). First we define (S, OP)

by $(S_1, OP_1) = B + (S, OP)$. Then we have:

$$
\begin{array}{ll}
K & = C + (S, h_1(OP), E_K) \\
E_K & = E_1 +_{E_B} E_C \\
h_{1S}(s) & = \begin{cases} \underline{if} & s \in S \\ \underline{then} & s \\ \underline{else} & d'(s) \end{cases} \\
h_{1OP}(op : s_1 \dots s_n \to s) & = \begin{cases} \underline{if} & (op : s_1 \dots s_n \to s) \in OP \\ \underline{then} & op : h_1(s_1) \dots h_1(s_n) \to h_1(n) \\ \underline{else} & d'(op) : d'(s_1) \dots d'(s_n) \to d'(s) \end{cases}
\end{array}
$$

Since d' is an inclusion $h_{1\Sigma}$ is an inclusion as well. This implies $(S_K, OP_K) = (S_2, OP_2)$. Together with $h_{1\Sigma} = d\Sigma$ and the fact that the construction of parameter passing diagrams uses an inclusion $p' : C \to SPEC_2$ we now have that (S_2, OP_2) is a pushout of the signatures in C and $SPEC_1$ over the signature of B. Since E_B is empty and E_C and E_2 are disjoint we have $E_K = E_2$. Since we know that h_{1E} and h_{2E} have to be inclusions and $K = SPEC_2$ we have that $SPEC_2$ is a pushout over $C \overset{d'}{\leftarrow} B \overset{i}{\rightarrow} SPEC_1$.

Given now another pushout (1) over the inclusion d as depicted below, where $B' = (S_{B'}, OP_{B'}, E_{B'})$, $C' = (S_{C'}, OP_{C'}, E_{C'})$ and j_1 is assumed to be a monomorphism.

$$
\begin{tikzcd}
B \arrow[rr, "i"] \arrow[dr, "\overline{b}"] \arrow[dddd, "d'"'] & & SPEC_1 \arrow[dddd, "d"] \\
& B' \arrow[ur, "j_1"'] \arrow[dd, "j_2"'] & \\
(2) & & (1) \\
& C' \arrow[dr, "j_3"] & \\
C \arrow[rr, "i'"'] \arrow[ur, "\overline{c}"] & & SPEC_2
\end{tikzcd}
$$

Applying theorem 5.2.2 on j_1 in diagram (1) implies that j_3 is a monomorphism as well. Since (1) is a pushout $d \circ j_1 = j_3 \circ j_2$ holds. Since d and j_1 are monomorphisms we have the monomorphisms property for $j_3 \circ j_2$ and so for j_2.

Lemma 5.2.1 $\forall b \in B \exists b' \in B' \bullet c_1(b') = b$

Proof: Assumed b is an operation symbol of OP_B. Then we have

- $\exists e \in E_2 \bullet e \notin E_1 \land b \in ops(e)$, by the construction of B.

- The pushout property of (2) delivers $\exists c' \in E_{C'} \bullet j_3(c') = e$.
- Since c_3 is a morphism in the category $SPEC$ we have $\exists op \in OP_{C'}$ such that $j_3(op) = b$.
- Now we have $b \in B \subseteq SPEC_1 \subseteq SPEC_2$ and $op \in C'$ with $j_3(op) = b$. That means $b \in SPEC_1$ and $op \in C'$ have a common image in $SPEC_2$. Since (1) is a pushout of injective morphisms we obtain $\exists b' \in B' \bullet j_1(b') = b$, following the pushout characterizations in [7].

Assumed b is a sort of S_B. By the construction of B_S the following case distinction is complete:

1. An operation symbol, having s in its type, is added along the inclusion d. Similar to the case above we can use the pushout property of (2) and that $j_3 \in SPEC_{Mor}$ to obtain

   $$\exists s \in C' \bullet j_3(s) = b$$

   And since (1) is a pushout diagram of monomorphisms we can conclude that s and and b have to be identified over B. So we have

   $$\exists b' \in B' \bullet j_1(b') = b.$$

2. An equation for an operation symbol f that has s in its type is added along the inclusion d. According to the case $b \in OP_B$ we have that there is an operation symbol op in B' such that $j_1(op) = f$. And since j_1 is a monomorphic specification morphism we have that there is a sort $b' \in S_{B'}$ such that $j_1(b') = b$.

Since the set of equations E_B is empty, the proof for the Lemma is completed. $\surd$

Lemma 5.2.2 $\forall c \in C \ \exists c' \in C' \bullet j_3(c') = c$

Proof:

- Given $c \in C$. By the construction of C it is known that the following case distinction is complete.
- First case: $c \in B$. In the proof for the Lemma above operation symbols and sorts $c' \in C'$ for that case were constructed such that $j_3(c') = b$. Choosing $b = c$ delivers a $c' \in C'$ with $j_3(c') = c$. $\surd$

- Second case: $c \in SPEC_2$ and $c \notin SPEC_1$. Since (1) is a pushout we obtain $\exists c' \in C' \bullet j_3(c') = c$. $\surd$

Knowing that j_1 and j_3 are injective we can define the partial specification morphisms $j_1^{-1} : SPEC_1 \nrightarrow B'$ and $j_3^{-1} : SPEC_2 \nrightarrow C'$.

$$j_1^{-1}(n) = \underline{if}\ \exists n' \in B' \bullet j_1(n') = n\ \underline{then}\ n'$$
$$j_3^{-1}(n) = \underline{if}\ \exists n' \in C' \bullet j_3(n') = n\ \underline{then}\ n'$$

By the two Lemmas above we can conclude that j_1^{-1} is defined for all $b \in B \subseteq SPEC_1$ and that j_3^{-1} is defined for all $c \in C \subseteq SPEC_2$. Moreover we have $j_1 \circ j_1^{-1} = id_{SPEC_1}$ and $j_3 \circ j_3^{-1} = id_{SPEC_2}$ and we know that j_1^{-1}, j_3^{-1} are injective. So we can compose the following specification morphisms.

1. $\overline{b} : B \to B'$ with $\overline{b} = j_1^{-1} \circ i$ is well defined and total since j_1^{-1} is defined for B and so for $i(B)$. Moreover, this morphism is injective since i and j_1^{-1} are injective. $\Rightarrow \overline{b} \in M_{inj}$

2. $\overline{c} : C \to C'$ with $\overline{c} = j_3^{-1} \circ i'$ is well defined and total since j_1^{-1} is defined for C and so for $i'(C)$. Moreover, this morphism is injective since i' and j_3^{-1} are injective. $\Rightarrow \overline{c} \in M_{inj}$

By defining the morphisms in that way we also obtain that $i = j_1 \circ \overline{b}$ and $i' = j_3 \circ \overline{c}$ because.

Now we have the same situation as in the case of the place/transition nets after defining $\overline{b}$ and $\overline{c}$. So the existence of initial pushouts over inclusions in the category **Spec** is proven. $\square$

Theorem 5.2.4 (Existence of Initial Pushouts II) *Given two specifications $SPEC_1$ and $SPEC_2$ with an injective specification morphism $d : SPEC_1 \to SPEC_2$. Then we get the specifications B and C with morphisms i, i', d' such that the diagram below is an initial pushout with respect to d.*

$$\begin{array}{ccc} B & \xrightarrow{i} & N_1 \\ \downarrow{\scriptstyle d'} & & \downarrow{\scriptstyle d} \\ C & \xrightarrow{i'} & N_2 \end{array}$$

Proof: This can be concluded from theorem 5.2.3 and the section A.2. $\square$

Theorem 5.2.5 (Algebraic Specification Pullbacks) *Given the specifications B, C, D with injective specification morphisms $b : B \to D, c : C \to D$. Now we have a specification A with morphisms $a_1 : A \to C$ and $a_2 : A \to B$ such that the diagram below becomes a pullback diagram.*

$$\begin{tikzcd} A \arrow[r, "a_2"] \arrow[d, "a_1"'] \arrow[dr, phantom, "(P)"] & B \arrow[d, "b"] \\ C \arrow[r, "c"] & D \end{tikzcd}$$

Theorem 5.2.6 (Inheritance of Monomorphisms under PB's) *If we have a pullback diagram as (P) above then we know if c and b are in class M_{inj} then also a_1 and a_2 are in class M_{inj}.*

Theorem 5.2.7 (Pushouts are Pullbacks) *If in the diagram (P) above (D, c, b) is a pushout of (A, a_1, a_2) and a_1, a_2 are injective specification morphisms then (A, a_1, a_2) is a pullback of (D, c, b).*

Theorem 5.2.8 (Pushout Pullback Decomposition) *Given a pushout (1)+(2) as depicted below, where (2) is assumed to be a pullback and the morphisms a_1, b_1, e, b_2, d are assumed to be injective. Now we have that (1) is a pushout.*

$$\begin{tikzcd} A \arrow[r, "a_2"] \arrow[d, "a_1"'] \arrow[dr, phantom, "(1)"] & B \arrow[r, "b_2"] \arrow[d, "b_1"'] \arrow[dr, phantom, "(2)"] & E \arrow[d, "e"] \\ C \arrow[r, "c"] & D \arrow[r, "d"] & F \end{tikzcd}$$

Remark: These theorems correspond to the assumptions 8, 7, 9 and 10. Their validity follows from the the fact that the category **Spec** is based on sets and the corresponding properties hold in **SET**. See [14] for detailed proofs.

So we have the following results. A component of algebraic specifications now consists of three specifications:

- a specification IMP, called import interface,
- a specification EXP, called export interface and
- a specification BOD, called body.

Moreover, the component contains an injective specification morphism $imp : IMP \to BOD$ and a **Spec** transformation $exp : EXP \Rightarrow BOD$.

The semantics are then given by the extension of all possible import transformations. Since the category **Spec** is a valid instantiation of the HLR transformation based component concept we have the compositionality of the semantics according to section 2.5.

5.3 Example

This small example specifies a total automaton. The export interface of the first component in this example, EXP_1, only knows the sorts $State$ and $String$ and an operation symbol run which is not specified at that point. Of course, this operation is meant to run as long as there are items to read in the input string and then to deliver the corresponding state of the automaton.

The body BOD_1 of the same component which extends EXP_1 then specifies the intended functionality of the operation symbol run. This is done by introducing and specifying operation symbols that can build strings from alphabet items. Because these operation symbols are then used to specify that run reads the string from left to right and reacts on each alphabet item in the string with a corresponding state change.

$COMP_1 = (IMP_1, EXP_1, BOD_1, imp_1, exp_1)$
$EXP_1 =$

$$
\begin{array}{ll}
\textbf{sorts} & State, String \\
\textbf{opns} & run : State\ String \to State \\
\textbf{eqns} & \emptyset
\end{array}
$$

$BOD_1 = EXP_1 +$

$$
\begin{array}{ll}
\textbf{sorts} & Alphabet \\
\textbf{opns} & step : State\ Alphabet \to State \\
 & empty : \to String \\
 & add : String\ Alphabet \to String \\
 & con : String\ String \to String \\
\textbf{eqns} & run(st, empty) = st \\
 & run(st, add(a, w)) = run(step(st, a), w) \\
 & con(empty, w) = w \\
 & con(w, empty) = w \\
 & con(add(a, w), l) = add(a, con(w, l))
\end{array}
$$

$IMP_1 =$

$$\begin{array}{ll} \textbf{sorts} & Alphabet, State \\ \textbf{opns} & step : State\ Alphabet \to State \\ \textbf{eqns} & \emptyset \end{array}$$

$exp_1 =$

$$\begin{array}{ccccc} EXP_1 & \longleftarrow & EXP_1 & \longrightarrow & BOD_1 \\ \downarrow & & \downarrow & & \downarrow \\ EXP_1 & \longleftarrow & EXP_1 & \longrightarrow & BOD_1 \end{array}$$

The morphism from the import to the body of $COMP_1$, imp_1, is an inclusion. The second component now specifies the concrete structure of the automaton, i.e. the operation symbol *step* is specified.

$COMP_2 = (IMP_2, EXP_2, BOD_2, imp_2, exp_2)$

$EXP_2 =$

$$\begin{array}{ll} \textbf{sorts} & Alphabet, State \\ \textbf{opns} & step : State\ Alphabet \to State \\ \textbf{eqns} & \emptyset \end{array}$$

$BOD_2 = EXP_2 +$

$$\begin{array}{ll} \textbf{opns} & start : \to State \\ & even : \to State \\ & odd : \to State \\ & 1 : \to Alphabet \\ & 2 : \to Alphabet \\ \textbf{eqns} & step(start, 1) = odd \\ & step(start, 2) = even \\ & step(even, 1) = odd \\ & step(even, 2) = even \\ & step(odd, 1) = even \\ & step(odd, 2) = odd \end{array}$$

$IMP_2 = (\emptyset, \emptyset, \emptyset)$

$exp_2 =$

$$\begin{array}{ccccc} EXP_2 & \longleftarrow & EXP_2 & \longrightarrow & BOD_2 \\ \downarrow & & \downarrow & & \downarrow \\ EXP_2 & \longleftarrow & EXP_2 & \longrightarrow & BOD_2 \end{array}$$

Here, the morphism from the import to the body is empty. If we now compose the two components via an identic transformation we obtain a specification component $COMP_3 = (IMP_2, EXP_1, BOD_3, imp_3, exp_3)$, where BOD_3, imp_3, exp_3 are defined according to the definition 2.4.2. This means that BOD_3 is computed within the following extension diagram.

$$
\begin{tikzcd}
EXP_1 = IMP_1 \arrow[r, "imp_1"] \arrow[d, Rightarrow, "exp_2"] & BOD_1 \arrow[d, Rightarrow, "exp'_2"] \\
BOD_2 \arrow[r, "imp'_1"] & BOD_3
\end{tikzcd}
\qquad
\begin{tikzcd}
EXP_1 = IMP_1 \arrow[r, "imp_1"] \arrow[d] \arrow[dr, phantom, "(1)"] & BOD_1 \arrow[d] \\
BOD_2 \arrow[r, "imp'_1"] & BOD_3
\end{tikzcd}
$$

Since in the direct transformation exp_2 the the image of the left side of the rule is equal to the context object of the transformation it is sufficient to compute the pushout (1). Below you see the body of the composed component.

$BOD_3 = EXP_1 +$
sorts $Alphabet$
opns $step : State\ Alphabet \to State$
$empty : \to String$
$add : String\ Alphabet \to String$
$con : String\ String \to String$
$start : \to State$
$even : \to State$
$odd : \to State$
$1 : \to Alphabet$
$2 : \to Alphabet$
eqns $run(st, empty) = st$
$run(st, add(a, w)) = run(step(st, a), w)$
$con(empty, w) = w$
$con(w, empty) = w$
$con(add(a, w), l) = add(a, con(w, l))$
$step(start, 1) = odd$
$step(start, 2) = even$
$step(even, 1) = odd$
$step(even, 2) = even$
$step(odd, 1) = even$
$step(odd, 2) = odd$

Below you see a figure of the automaton that is specified by $COMP_3$. The idea of this automaton is to run it from the state *start* with strings that are in $\{1,2\}^*$. Then the automaton delivers the state *odd* when the sum of the digits in the string is odd and the state *even* if the opposite is true.

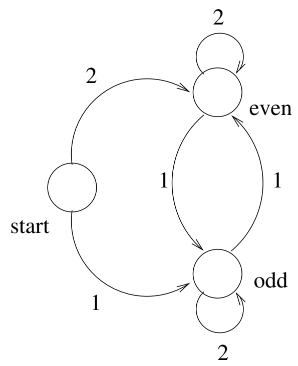

Figure 5.1: Automaton specified by $COMP_3$

Chapter 6

Instantiation to Algebraic High-Level Nets

This chapter gives an idea of how to instantiate the component concept for HLR categories with algebraic high-level nets, which are an integrated formalism providing means to describe behavior and data structures. Since algebraic high-level nets have strong theoretic foundations they are a common technique in security relevant software engineering. These nets can be understood as petri nets whose tokens are complex algebra elements belonging to a corresponding algebraic specification. The terms over the signature of this specification can be used to express additional equations that have to be valid for the firing of selected transformations.

6.1 Introduction to Algebraic High-Level Nets

Since there are different notions of algebraic high-level nets it is important to point out that the definitions below correspond to the nets in [22].

Definition 6.1.1 (Algebraic High-Level Net) *An* algebraic high-level net

$$AN = (SPEC, P, T, pre, post, cond, A)$$

is defined by: a signature of sorts and operations $\Sigma = (S, OP)$*, equations* E *over* Σ *leading to an algebraic specification* $SPEC = (S, OP, E)$*, a set of places* P*, a set of transitions* T*, two mappings* $pre, post : T \to (T_{OP}(X) \times P)^{\oplus}$*, a mapping* $cond : T \to \mathcal{P}_{fin}(EQNS(\Sigma))$ *and a* $SPEC$*-algebra* A*.*

For sake of simplicity, in this paper a special version of morphisms for AHL nets is used where the specification part and the algebra are not changed along the morphisms.

Definition 6.1.2 (Algebraic High-Level Net Morphisms) *An algebraic high-level net morphism $f : AN_1 \to AN_2$ consists of:*

- *a function $f_P : P_1 \to P_2$ and*
- *a function $f_T : T_1 \to T_2$,*

where we assume $AN_i = (SPEC, E, P_i, T_i, pre_i, post_i, cond_i, A)$ for $i \in \{1, 2\}$ and the diagram below has to commute for the two pre and the two post functions and the two cond functions.

$$\begin{tikzcd}
 & T_1 \arrow[dl, "cond_1"'] \arrow[r, shift left, "pre_1"] \arrow[r, shift right, "post_1"'] \arrow[dd, "f_T"] \arrow[ddr, phantom, "="] & (T_{OP}(X) \times P_1)^{\oplus} \arrow[dd, "(id \times f_P)^{\oplus}"] \\
\mathcal{P}_{fin}(EQNS(\Sigma)) \arrow[r, phantom, "="] & {} & \\
 & T_2 \arrow[ul, "cond_1"] \arrow[r, shift left, "pre_2"] \arrow[r, shift right, "post_2"'] & (T_{OP}(X) \times P_2)^{\oplus}
\end{tikzcd}$$

Remark: The algebraic high-level nets and the corresponding morphisms together make up the category **Ahl-Net**.

The generic component concept based on HLR transformations can be instantiated with the category **Ahl-Net** of algebraic high-level nets such that the extension property and horizontal and vertical compositionality as well as decompositionality of extension diagrams are satisfied for the class M of injective **Ahl-Net** morphisms. This means that we have a well defined transformation semantics for **Ahl-Net** components, composition of components and compositionality of transformation semantics. In contrast to the previous two instantiations the detailed proof is omitted in this case.

It is well known that the category **Ahl-Net** has pushouts which are constructed componentwisely (see [22]). Moreover, the gluing condition for the existence of pushout complements ([22]) implies the existence of initial pushouts for injective net morphisms. The other assumptions needed for the instantiation of the component framework can be shown because all pushout and pullback constructions in **Ahl-Net** can be done componentwisely.

6.2 Example

Next, a sample component from the software model of our approach to the case study introduced in section 4.3 is presented. In order to be more precise and to be able to specify the data part algebraic high-level nets are used in the components for this part.

The intention of the domain model was to specify the behavior of the entire system and its environment. The software model now focuses on the software design for the H-AGVs. As usual the component structure of the software model (Figure 6.1) is different to the one of the domain model (Figure 4.2). It is outside the scope of this report to discuss how the two models are related. Note that the components in Figure 6.1 model the software for one H-AGV and not for the whole system. The components

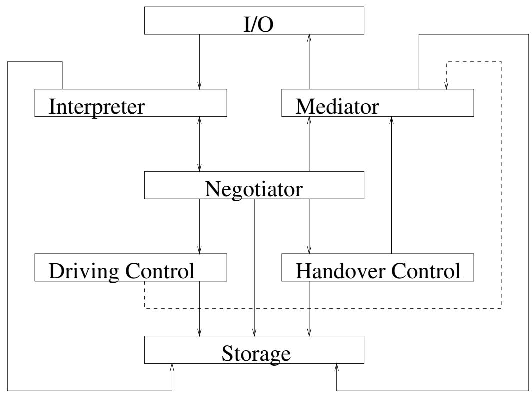

Figure 6.1: Overview of software model components

have the following tasks within the system.

- The *I/O* component specifies how messages are sent and received and recognized as answers on sent messages.
- The *Interpreter* component contains the specification of the interpretation of incoming messages that are not an answer for a previously sent message.
- The *Mediator* component offers the specification of the creation of outgoing messages.
- The *Negotiator* component specifies how an H-AGV acts in the case of an negotiation.

- The *Driving Control* component describes how an H-AGV drives. This includes the calculation of the best route and the reservation of this route.
- The *Handover Control* component specifies how workpieces are put and taken to/from the buffer of a machine tool.

As sample component to be presented in this paper the *I/O* component is chosen. The *I/O* component in Figures 6.2–6.4 specifies the communication protocol for the H-AGVs. Since the machine tools use the same protocol this component could be reused for the specification of the machining tools. In the export of this component we have an algebraic high-level net with a single transition. This transition takes a tuple of data elements from the place *out buffer* – the first element contains the content of the message to send, the second one says whether this message is an answer to a message received before and the next elements optionally contain the identification number of the message this one is an answer for and the identification number of the receiver. If there are any answers to a sent message then these answers are put to the place *result*.

Following the ideas of the framework the body now models the realization of this behavior. The export and the body are related by a transformation. Similar to Figure 4.9 above the transformation is equal to the corresponding rule. Both are not shown explicitly here. The left hand side of the rule is equal to the export of the *I/O* component, the interface consists of the places *out buffer* and *result*, and the right hand side of the rule is equal to the body of the *I/O* component. The transformation deletes the transition *send sync* from the export net but keeps the places *out buffer* and *result* and adds the other parts of the net in Figure 6.3.

The transition *send* again takes the four data elements from the *out buffer*. Moreover, this transition takes an identification number from the place *id_buffer_1* and saves it to the places *id_buffer_2* and *sent buffer*. This number is then the identification number of the new message. In order to distinguish the sent messages this number is increased after each sent message. The place *broadcast medium* is meant to model a corresponding physical device. This place is shared by all H-AGVs. This could be expressed with the union operation mentioned in Chapter 7. The message that finally reaches the *broadcast medium* contains the four elements taken from *out buffer* plus the id of the H-AGV (*self_id*) and the id of the new message (*m_id*).

The body of the *I/O* component also contains two transitions that model how messages are received. There is no abstract representation of this feature in the export. The transition *receive answer* reads all messages whose reference id is equal to one of the ids on the place *sent buffer* and that are adressed to the H-AGV with the identification number *self_id*.

All other messages have to be interpreted by a corresponding *Interpreter* component. As a consequence the body of the *I/O* component contains a transition *interpret* which is part of the net in the import of the *I/O* component. This import is connected to the export of the *Interpreter* component (see Figure 6.1).

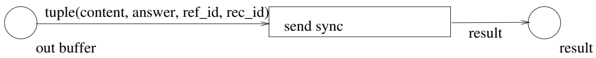

Figure 6.2: Export of component *I/O*

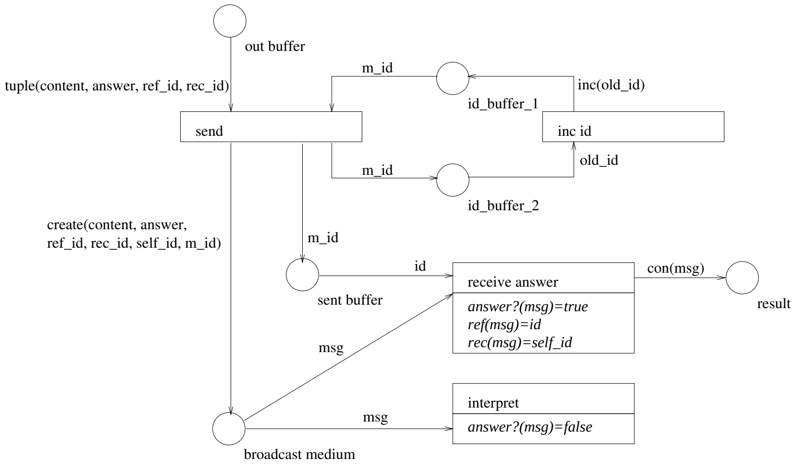

Figure 6.3: Body of component *I/O*

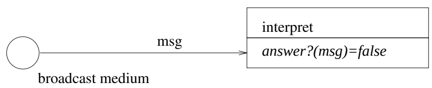

Figure 6.4: Import of component *I/O*

All three nets in the component *I/O* are based on the following algebraic specification, a suitable *SPEC*-algebra *A* and additional variables used in the inscriptions of the arcs and in the conditions of the transitions.

$$
\begin{array}{ll}
\multicolumn{2}{l}{SPEC = BOOL+} \\
\textbf{sorts} & triple, message, content, id \\
\textbf{opns} & tuple : content\ bool\ id\ id \to triple \\
& create : content\ bool\ id\ id\ id\ id \to message \\
& con : message \to content \\
& get\_id : message \to id \\
& answer? : message \to bool \\
& ref : message \to id \\
& rec : message \to id \\
& sender : message \to id \\
& inc : id \to id \\
& self\_id : \to id \\
\textbf{vars} & X_{message} = \{msg\} \quad X_{content} = \{c, content\} \\
& X_{bool} = \{a, answer\} \\
& X_{id} = \{i1, i2, i3, i4, ref\_id, rec\_idm\_id, old\_id, id\} \\
\textbf{eqns} & con(create(c, a, i1, i2, i3)) = c \\
& get\_id(create(c, a, i1, i2, i3, i4)) = i4 \\
& answer?(create(c, a, i1, i2, i3, i4)) = a \\
& ref(create(c, a, i1, i2, i3, i4)) = i1 \\
& sender(create(c, a, i1, i2, i3, i4)) = i2 \\
& rec(create(c, a, i1, i2, i3, i4)) = i3
\end{array}
$$

Chapter 7

Conclusion

This report delivers the theoretical background for the instantiation of an abstract transformation based component framework to HLR transformations. This was achieved by showing the extension property for HLR systems and based on that the horizontal and vertical compositionality of HLR extension diagrams and the horizontal and vertical decomposition of HLR extension diagrams. The resulting component concept for HLR categories was then instantiated to sample categories and classes of morphisms. This was done by showing the different assumptions for the concrete categories and classes of morphisms. But there are still some known HLR categories left to examine, e.g. the category of abstract statecharts ([3]).

The instantiation to HLR transformations could be further refined with respect to the set of allowed transformations to connect components and to refine the export interface by the body. It seems very senseful to restrict this set to transformations that preserve certain properties of the corresponding specifications. Especially the possibility to delete system parts without adding adequate substitutions should be forbidden. Since the transformation semantics is meant to be a semantics of possible refinements the allowed transformations should be constrained to refinements. At this point it is not yet clear how to formulate such transformation constraints. There are two main ways.

- The first way would be to define properties of transformations in categorical terms. On the one hand this would not cause further work in the underlying HLR categories but on the other hand it seems difficult to find such an abstract formulation of the wanted properties.

- The second possibility is to restrict the transformations in a reasonable way in the underlying HLR categories. On this way it should be possible to define very concrete constraints on the transformations, as for example the *Q-Transformations* in [23].

At both approaches it is possible to distinguish between descriptive constraints, as for example to forbid to create *dangling edges* along a transformation of graphs, and between more precise constraints as for example the boundary based consistency property. Of course, both approaches have their own advantages.

In [11] a first formulation of components with multiple interfaces was introduced and discussed with respect to an instantiation to HLR systems. The main idea of this extension is to have components with several import and export interfaces together with the possibility to connect a single import and a single export interface. This new kind of composition is then called *partial composition*. In Figure 6.1 in Section 6.2 the arrows between the components can be understood as the transformations for corresponding partial compositions.

This extension would give more power to the presented component concept because it would allow to specify much more structured component architectures. See [17] for a more detailed discussion of the advantages of components with multiple interfaces and an UML based definition of such a component framework.

The multiple interface extension would also allow to express *architectural connectors* in the sense of [15]. More exactly, a connector would be realized by a component, where the roles of the connector would be given by the family of import interfaces. The export interface of this component would be empty since connectors do not offer any functionality to their environment. At this point it might be useful to collect the different export interfaces of the components that realize the roles. By now the definition of the partial composition proposed in [11] closes that interfaces.

Another extension of the component framework proposed in [11] is a union operation in the sense of a gluing of components over a shared subpart – similar to the union operation on algebraic module specifications in the sense of [9]. A senseful application of this feature was already discussed in Section 6.2.

Bibliography

[1] A. Braatz and A. Ritter. Referenzfallstudie Produktionstechnik, available at tfs.cs.tu-berlin.de/projekte/indspec/spp/themenbereiche.html, 2001.

[2] J. Cheesman and J. Daniels. *UML Components*. Addison-Wesley, 2001.

[3] H. Ehrig, M. Gajewsky, and J. Padberg. Action nets and abstract statecharts in the theory of high-level replacement systems. In *Proc. of First European GETGRATS Workshop*, October 1997.

[4] H. Ehrig, M. Gajewsky, and F. Parisi-Presicce. High-level replacement systems applied to algebraic specifications and petri nets. In *Handbook of graph grammars and computing by graph transformation: vol. 3: concurrency, parallelism, and distribution*, pages 341–399. World Scientific Publishing Co., Inc., 1999.

[5] H. Ehrig and M. Große-Rhode. *Informatik, Forschung und Entwicklung*, volume 16, chapter Integration von Techniken der Softwarespezifikation für Ingenieurwissenschaftliche Anwendungen, pages 100–117. Springer Verlag, June 2001.

[6] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. volume 1 of *Math. Struct. in Comp. Science*, pages 361–404. Cambridge Univ. Press, 1991.

[7] H. Ehrig and H.-J. Kreowski. Contributions to the algebraic theory of graph grammars. Technical Report 76-22, TU-Berlin, 1976.

[8] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theor. Comp. Science*. Springer, 1985.

[9] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2: Module Specifications and Constraints*, volume 21 of *EATCS Monographs on Theor. Comp. Science.* Springer, 1990.

[10] H. Ehrig, B. Mahr, F. Cornelius, M. Große-Rhode, and F. Zeitz. *Mathematisch-Strukturelle Grundlagen der Informatik.* Springer-Verlag, second edition, 2001.

[11] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A component framework based on high-level replacement systems. In *Proc. of the Int. Workshop on Graph Transformation and Visual Modeling Techniques*, volume 72 No. 3 of *Electronic Notes in Theoretical Computer Science*. Elsevier Sciences, 2002.

[12] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A generic component framework for system modeling. In *Proc. FASE*, volume 2306 of *LNCS*, pages 33–48. Springer, 2002.

[13] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A transformation-based component framework for a generic integrated modeling technique. In *Proc. of Integrated Design and Process Technology 2002.* Society for Design and Process Science, 2002.

[14] H. Ehrig and F. Parisi-Presicce. Nonequivalence of categories for equational algebraic specification in view of high-level-replacement systems. Technical Report 91-16, TU-Berlin, 1991.

[15] J.L. Fiadeiro and A. Lopes. Semantics of architectural connectors. In *TAPSOFT'97, M.Bidoit and M.Dauchet (eds)*, volume 1214 of *LNCS*, pages 505–519. Springer-Verlag, 1997.

[16] M. Große-Rhode, R.-D. Kutsche, and F. Bübl. Concepts for the evolution of component-based software systems. Technical Report 2000/11, TU-Berlin, 2000.

[17] L. Iribarne, J.M. Troya, and A. Vallecillo. Selecting software components with multiple interfaces. In *Proceeding of the 28th EUROMICRO Conference – Component-Based Software Engineering*, pages 26–32. IEEE Computer Society. Dortmund, Germany, 2002.

[18] M. Klein, M. Özhan, and M. Piirainen. Iosip case study model files, available at tfs.cs.tu-berlin.de/~mesut/iosip/, 2002.

[19] J. Padberg. Survey of high-level replacement systems. Technical Report 93-8, TU-Berlin, 1993.

[20] J. Padberg. *Abstract Petri Nets: A Uniform Approach and Rule-Based Refinement.* PhD thesis, Technical University of Berlin, Dept. of Comp. Sci. , Shaker Verlag, 1996.

[21] J. Padberg. Place/transition net modules: Transfer from specification modules. Technical Report 2001-03, TU-Berlin, 2001.

[22] J. Padberg, H. Ehrig, and L. Ribeiro. *Math. Struct. in Comp. Science*, volume 5, chapter Algebraic High-Level Net Transformation Systems, pages 217–256. Cambridge Univ. Press, 1995.

[23] J. Padberg, M. Gajewsky, and C. Ermel. Rule-based refinement of high-level nets preserving safety properties. *Lecture Notes in Computer Science*, 1382:221–238, 1998.

[24] J. Padberg and G. Taentzer. Embedding of derivations in high-level replacement systems. Technical Report 93-9, TU-Berlin, 1993.

[25] C. A. Petri. *Kommunikation mit Automaten.* PhD thesis, Schriften des Instituts für instrumentelle Mathematik, Bonn, 1962.

Appendix A

Used Categorical Theorems

This appendix covers some important categorical theorems that have been used in the proofs of the report. Of course, this is only a selection of important theorems that are not part of standard introductions for category theory, as for example [10].

A.1 Pushout Characterization

This section repeats some main results of [7]. Originally, the following theorem was only meant for the category of graphs. But since the proof only uses the fact that graphs are based on sets, it is no problem to apply the results of the theorem on the category **SET**. Before formulating the main theorem, some definitions have to be introduced.

Definition A.1.1 *A morphism* $g_1 : B_1 \to C$ *is called* (source) injective up to $f_1 : A \to B_1$ *if* $\forall x, x' \in B_1$ *with* $g_1(x) = g_1(x')$ *we have* $x, x' \in f_1(A)$.

Definition A.1.2 *Given a commutative diagram as depicted below. Now* (g_1, g_2) *satisfies the* reduced chain condition *if* $\forall b_1 \in B_1, b_2 \in B_2$ *with* $g_1(b_1) = g_2(b_2)$ *there is an* $a \in A$ *such that* $f_1(a) = b_1$ *and* $f_2(a) = b_2$.

Definition A.1.3 *The pair* (g_1, g_2) *is* epimorphic *if* $g_1(B_1) \cup g_2(B_2) = C$.

Theorem A.1.1 *Given a commutative diagram in the category* **SET** *as depicted below.*

$$\begin{tikzcd} A \arrow[r, "f_1"] \arrow[d, "f_2"'] & B_1 \arrow[d, "g_1"] \\ B_2 \arrow[r, "g_1"] & C \end{tikzcd}$$

Now this diagram is a pushout, if:

1. *g_1 is injective up to f_1,*
2. *g_2 is injective up to f_2,*
3. *(g_1, g_2) is epimorphic,*
4. *the diagram satisfies the reduced chain condition.*

The proof and a different version of the pushout characterization can be found in [7]. Since the sample categories in the chapters 4 and 5 are based on sets and mappings and the corresponding pushouts are based on pushouts in **SET** we can conclude that the theorem valid for the categories **P/T** and **Spec**. In that case the needed properties for a pushout have to be shown componentwisely.

A.2 Extension of Initial Pushouts

Theorem A.2.1 (Extension of Initial Pushouts) *Given a category* **CAT** *and a class of morphisms M_1 such that $M_1 \subseteq$* $\mathbf{CAT_{Mor}}$. *Assumed we have the existence of initial pushouts in that category for morphisms in M_1 then we have it for the class M_2 that contains the morphisms of M_1 composed with* **CAT***-isomorphisms.*

Proof: Given a morphism $d : G_1 \to G_2$ in M_2. Now we know that we can decompose this morphism into a morphism $d_1 : G_1 \to G_1'$ such that $d_1 \in M_1$ and an isomorphism $d_2 : G_1' \to G_2$, such that $d_2 \circ d_1 = d$. Per assumption we can construct the initial pushout (I$_1$) over the M_1-morphism d_1 as depicted below, such that $i \in M_1$. This implies that $i \in M_2$.

$$\begin{tikzcd}
B \arrow[r, "i"] \arrow[d, "d_1'"'] \arrow[dr, phantom, "(I_1)"] & G_1 \arrow[d, "d_1"] \\
C \arrow[r, "i'"] & G_1' \arrow[d, "d_2"] \\
 & G_2
\end{tikzcd}$$

Since d_2 is an isomorphism the following diagram is a pushout as well.

$$\begin{tikzcd} B \arrow[r, "i"] \arrow[d, "d_1'"'] \arrow[dr, phantom, "(I_2)"] & G_1 \arrow[d, "d_2 \circ d_1 = d"] \\ C \arrow[r, "d_2 \circ i'"] & G_2 \end{tikzcd}$$

Now given a comparative pushout over the M_2-morphism d as in the diagram (1) below, where j_1 is assumed to be a monomorphism.

$$\begin{tikzcd} B' \arrow[r, "j_1"] \arrow[dd, "j_2"'] & G_1 \arrow[d, "d_1"] \\ \arrow[r, phantom, "(1)"] & G_1' \arrow[d, "d_2"] \\ C' \arrow[r, "j_3"] & G_2 \end{tikzcd} \qquad \begin{tikzcd} B' \arrow[r, "j_1"] \arrow[d, "j_2"'] \arrow[dr, phantom, "(2)"] & G_1 \arrow[d, "d_1"] \\ C' \arrow[r, "j_3'"] & G_1' \end{tikzcd}$$

We can now derive a comparative pushout (2) for the M_1-morphism d_1. Let $j_3' : C' \to G_1'$ be defined as $j_3' = d_2^{-1} \circ j_3$ (This is possible because d_2 is an isomorphism). Now we have by the uniqueness of pushouts and by the fact that G_1' and G_2 are isomorphic that diagram (2) is a pushout as well.

At this point we use the initial property of the pushout (I). By that we obtain unique M_1-morphisms $\overline{b} : B \to B'$ and $\overline{c} : C \to C'$, such that:

- $i = j_1 \circ \overline{b}$ and
- $i' = j_3' \circ \overline{c}$ and
- $B' \overset{j_2}{\to} C' \overset{\overline{c}}{\leftarrow} C$ is a pushout over $B' \overset{\overline{b}}{\leftarrow} B \overset{d_1'}{\to} C$.

$$
\begin{tikzcd}
B \arrow[rr, "i"] \arrow[dr, "\overline{b}"'] \arrow[ddd, "d_1'"'] & & G_1 \arrow[ddd, "d_1"] \\
 & B' \arrow[ur, "j_1"'] \arrow[d, "j_2"'] & \\
 & C' \arrow[dr, "j_3'"] & \\
C \arrow[ur, "\overline{c}"] \arrow[rr, "i'"'] & & G_1' \arrow[d, "d_2"] \\
 & & G_2
\end{tikzcd}
$$

Note that $\overline{b}$ and $\overline{c}$ are also in class M_2. We have to show that $j_3 \circ \overline{c} = d_2 \circ i'$.

$$
\begin{array}{rll}
 & j_3 \circ \overline{c} & \\
= & d_2 \circ d_2^{-1} \circ j_3 \circ \overline{c} & d_2 \text{ is isom.} \\
= & d_2 \circ j_3' \circ \overline{c} & \text{definition } j_3' \\
= & d_2 \circ i' & \text{construction of } \overline{c}
\end{array}
$$

Since we have $j_3 = d_2 \circ j_3'$ the uniqueness of $\overline{b}$ and $\overline{c}$ also holds in the context of diagram (1), which implies that (I_2) is an initial pushout. $\square$