Reconfigurable Petri Systems with Negative Application Conditions

Diploma Thesis

Alexander Rein

Bericht–Nr. 2008/01
ISSN-Nummer: 1436-9915

Acknowledgment

I would like to thank PD Dr. habil. Julia Padberg, Leen Lambers and Ulrike Prange for their professional support. In addition, special thanks goes to my lovely girlfriend Sabrina Jury for her big patience and her very big support in numerous areas. Moreover, I would also like to thank my great parents for supporting me in every circumstance. Finally, I am very grateful to Conny Ullrich for the idea of the airport case study.

Danksagung

Ich danke PD Dr. habil. Julia Padberg, Leen Lambers und Ulrike Prange für ihre fachliche Unterstützung. Außerdem gilt mein spezieller Dank meiner lieben Freundin Sabrina Jury für ihre große Geduld und ihre sehr große Unterstützung in zahlreichen Bereichen. Weiterhin bedanke ich mich auch bei meinen großartigen Eltern für ihre Unterstützung in jeder Lebenslage. Schließlich bin ich Conny Ullrich sehr dankbar für die Idee der Flughafen-Fallstudie.

Abstract

This thesis introduces negative application conditions (NACs) for varied kinds of reconfigurable Petri systems. These are Petri systems together with a set of transformation rules that allow changing the Petri system dynamically. Negative applications are a control structure for restricting the application of a rule if a certain structute is present.
As introduced in [Lam07] and [LEOP08], (weak) adhesive high-level replacement (HLR) categories with negative application conditions are (weak) adhesive HLR categories with three additional distinguished morphism classes and some additional properties. These properties are required for generalizing results like Local Church-Rosser Theorem, Parallelism Theorem, Completeness Theorem of Critical Pairs, Concurrency Theorem, Embedding and Extension Theorem and Local Confluence Theorem for the use of negative application conditions. The main goals of this thesis are proving that the categories **PTSys** of P/T systems, **AHLNet** of algebraic high-level (AHL) nets, **AHLSystems** of AHL systems and **PTSys**(**L**) of L-labeled P/T systems are weak adhesive HLR categories with negative application conditions. Therefore, these categories are formally introduced and the required properties are proven in detail. Additionally, the practical application of the achieved results is presented in form of case studies.

Keywords: Petri net, Petri system, P/T net, P/T system, AHL net, AHL system, labels, net transformation, control structure, negative application condition, adhesive HLR category with NACs, adhesive HLR system with NACs, case study, airport control system, ACS

Zusammenfassung

Diese Arbeit führt negative Anwendungsbedingungen (NACs) für verschiedene Typen von rekonfigurierbaren Petri Systemen ein. Dies sind Petri Systeme mit einer Menge von Transformationsregeln, die eine dynamische Veränderung des Petri Systems ermöglichen. Negative Anwendungsbedingungen sind eine Kontrollstruktur um die Anwendung einer Regel zu verbieten, wenn eine bestimmte Struktur vorhanden ist.
Wie in [Lam07] und [LEOP08] vorgestellt, sind schwach adhäsive HLR Kategorien mit negativen Anwendungsbedingungen schwach adhäsive HLR Kategorien mit drei zusätzlichen, ausgezeichneten Morphismenklassen und einigen zusätzlichen Eigenschaften. Diese Eigenschaften werden benötigt, um Ergebnisse wie das Lokale Church-Rosser Theorem, das Parallelismustheorem, das Vollständigkeitstheorem der kritischen Paare, das Nebenläufigkeitstheorem, das Einbettungs- und das Erweiterungstheorem und das Lokale Konfluenz Theorem für die Benutzung mit negativen Anwendungsbedingungen zu verallgemeinern. Das Hauptziel dieser Arbeit besteht darin nachzuweisen, dass die Kategorien **PTSys** der P/T Systeme, **AHLNet** der AHL Netze, **AHLSystems** der AHL Systeme und **PTSys**(**L**) der L-gelabelten P/T Systeme schwach adhäsive HLR Kategorien mit negativen Anwendungsbedingungen sind. Dafür werden diese Kategorien formal eingeführt und die dafür benötigten Eigenschaften detailliert bewiesen. Zusätzlich wird die praktische Anwendung der erzielten Ergebnisse in Form von Fallstudien dargelegt.

Schlüsselwörter: Petri Netz, Petri System, P/T Netz, P/T System, AHL Netz, AHL System, Labels, Netz Transformation, Kontrollstruktur, negative Anwendungsbedingung, adhäsive HLR Kategorie mit NACs, adhäsives HLR System mit NACs, Fallstudie, Flughafen Kontrollsystem, ACS

Contents

Nomenclature

ACS	airport control system
AGG	Attributed Graph Grammar System
AHL	algebraic high-level
AHO	algebraic higher-order
Def.	Definition
DP	dangling points
DPO	double-pushout
e.g.	exempli gratia (for example)
epi	epimorphism
GP	gluing points
HLR	high-level replacement
i.e.	id est (that is)
IP	identification points
MANET(s)	mobile ad-hoc network(s)
mono	monomorphism
NAC(s)	negative application condition(s)
P/T	place / transition
PB	pullback
PO	pushout
resp.	respectively
RON(s)	reconfigurable object net(s)

TFS ..... *Theoretische Informatik und Formale Spezifikationen*
(Theoretical Computer Science and Formal Specifications)

VKS ..... van Kampen square

Chapter 1

Introduction

1.1 Overview and Motivation

Petri nets are a well-known formal modeling technique in theoretical computer science. Their application area ranges over the description of workflows, the simulation of concurrent and sequential processes, the mathematical representation of distributed systems and the modeling of interactive web applications and complex object-oriented software. Moreover, Petri nets have numerous application areas beyond computer science like e.g. the modeling and simulation of manufacturing systems and assembly processes and, in industrial automation, the description of the control of industrial machinery and processes. Petri nets were invented in 1962 by Carl Adam Petri in the form of *place / transition nets* (short *P/T nets*). Advantages of P/T nets are their comprehensible graphical notation as bipartite and directed graphs consisting *places* and *transitions* as nodes (see Figure 1.1) on the one hand and their mathematical foundation on the other hand. Places may contain any number of so called *tokens*. A distribution of tokens over all places of a P/T net is called *marking*. A transition is *enabled* if there are (enough) tokens at all input places. If a transition is enabled it is able to *fire*, i.e. it removes a specified number of tokens of the input places and puts a specified number of tokens at the output places. Firing of transitions is nondeterministic. Since the invention of P/T nets a variety of mathematical descriptions of P/T nets have been established. In this thesis, the well-known monoid notation presented in [MM90] is used.

A P/T system is the extension of a P/T net by an initial marking. P/T systems

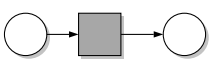


Figure 1.1: P/T Net PN

are often prefered for modeling since the initial situation (e.g. the initial state of a system) can be expressed by the initial marking.

Since the etablishment of Petri nets in the sixties of the last century, a lot of extensions of Petri nets were introduced. One of these extensions are *algebraic high-level nets* (short *AHL nets*). An AHL net is the extension of a P/T net by data types. Therefore, an algebraic specification and an algebra are added to the net. Tokens are now data types in contrast to P/T nets where tokens are indistinguishable. Every place has a determined type and can only contain tokens of this type. Additionally, transitions have fire conditions and are only enabled if all fire conditions are satisfied. Analogous to the case without data types, an *algebraic high-level system* (short *AHL system*) is the extension of an AHL net by an initial marking. Modeling with data types provides the advantage that the underlying net structure is usually much smaller than the corresponding net without data types. Hence, AHL nets are a very useful extension of P/T nets.
The phrase *Petri net* is often used as synonym for *P/T net*. In this thesis, *Petri net* stands for low-level (i.e. P/T) as well as for high-level (i.e. AHL) net.
The relationship between two Petri nets can be described through a *Petri net morphism*. Petri net morphisms are tuples of functions mapping the single components of a Petri net with some special properties introduced later in this thesis. Petri net morphisms can be extended to *Petri morphisms* describing the relationship between Petri systems. Petri morphisms are Petri net morphisms with an additional property with respect to the marking of the nets. They are also introduced later in this thesis.

A reconfigurable Petri system is a Petri system with a set of transformation rules for transforming this Petri system. Reconfigurable Petri systems are a suitable visual language for describing dynamical systems, i.e. systems which are able to adapt to changes while running. Thereby, the Petri system describes the behaviour of the system and the transformations describe the adaptation of the system to changes. These transformations base on the *algebraic approach* presented in [EEPT06]. This approach is based on *pushout* constructions. Pushouts are a concept of *category theory*. This is a mathematical fundamental theory dealing in an abstract way with mathematical structures and the relationships between them. A detailed introduction to category theory can be found in [AHS90]. In fact, there are two main variants of the algebraic approach, the *single-* and the *double-pushout (DPO) approach*. In this thesis, the DPO approach introduced in [EEPT06] is used. Since the algebraic approach is based on pure categorical constructions, it has a lot of instantiations, for example graphs, labeled graphs, typed graphs, attributed graphs, hypergraphs, Petri nets, Petri systems but also non-visual instantiations like algebraic specifications. The concept of these so called *high-level replacement (HLR) systems* introduced in [EHKPP91] and [EHKP91] was joined to that of *adhesive categories* introduced by Lack and Sobociński in [LS04], leading to the concept of *(weak) adhesive HLR categories* described in Chapters 4 and 5 in [EEPT06]. A (weak) adhesive HLR category is a category with special properties, which are not as strict as the properties of adhesive categories. This means, every adhesive category is also a (weak) adhesive HLR category, but the inverse conclusion does not hold in general. (Weak) adhesive HLR categories base on a distinguished subclass of monomorphisms $\mathcal{M}$ with some

special properties instead on the class of all monomorphisms as adhesive categories. There are adhesive HLR categories and weak adhesive HLR categories. The second mentioned are adhesive HLR categories with some weakened properties. The notion (weak) adhesive HLR category means that either an adhesive HLR or a weak adhesive HLR category can be considered. Some of the most significant results for adhesive HLR systems are the *Local Church-Rosser Theorem*, the *Parallelism Theorem*, the *Concurrency Theorem*, the *Embedding* and the *Extension Theorem*, the *Completeness Theorem of Critical Pairs* and the main result the *Local Confluence Theorem*. All these theorems are introduced, described in detail and proven in [EEPT06]. Simplified and informally, the *Local Church-Rosser Theorem* states that the order of two sequential transformations with a special property, called *independence*, of one object does not matter (see Figure 1.2) and the *Parallelism Theorem* states the existence of a direct transformation performing both transformations in one step. The existence of a direct transformation for every pair of transformations (even *dependent* transformations) is guaranteed by the *Concurrency Theorem*. The *Embedding* and the *Extension Theorem* allow extending a transformation into a larger context. The *Completeness Theorem of Critical Pairs* expresses that every pair of *parallel dependent* transformations belongs to a *critical pair*. Critical pairs are required for the *Local Confluence Theorem*. The *Local Confluence Theorem* states some conditions for the local confluence of the transformation system. Local confluence means that for all pairs of direct transformations, it holds that the different results can be transformed to the same result. This property is required for most transformation systems. These theorems hold for all instances of adhesive HLR systems, although some of them require some special properties.

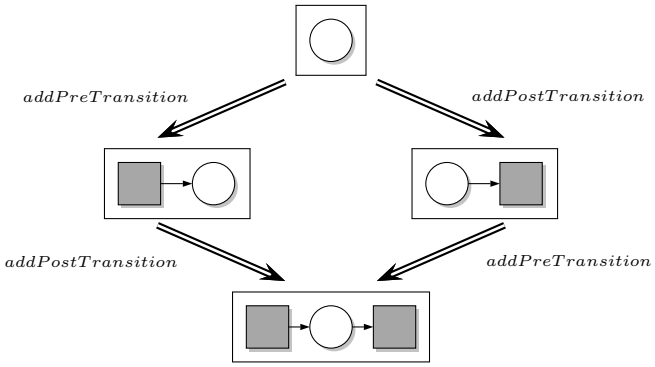


Figure 1.2: Local Church-Rosser Property

In most transformation systems, there are a lot of basic conditions for the applicability of transformations. For example, a transformation may not be applied in some situations, although it is theoretically possible. For expressing conditions like this, some kind of *control structure* for the applicability of rules is required. Control structures are necessary if rules should be applied automated and also reasonable if

the rules should be applied manually for preventing human failures.
Negative application conditions (*NACs*), introduced in Chapter 7 in [EEPT06], are such a control structure for adhesive HLR systems. They restrict the application of a rule if a certain structure is present. Furthermore, there are other control structures like labels, presented later in this thesis, and transformation units, presented in [KKS97], which are not discussed in this thesis.
Two kinds of negative application conditions can be distinguished. On the one hand there are *left negative application conditions* and on the other hand there are *right negative application conditions*. The first mentioned restrict the application of a rule if a certain structure is present before applying the rule and the second mentioned if a certain structure is present after applying the rule. However, every right NAC can be transformed to an equivalent left NAC as shown in Lemma 2.11 in [LEOP08]. For this reason, only left negative application conditions are considered in this thesis and they are simply called *negative application conditions* in the following.
The necessity of negative application conditions for transformation systems is obvious. A simple example therefore is the restriction of the repeated application of a rule. This is shown in Figure 1.3), where a rule *addPreTransition* is applied repeatedly to a P/T system. Expanding this rule by an adequate negative application condition forbids transformation (2) and all following transformations shown in Figure 1.3 since the structure of the NAC can be found in the second P/T system. Of course, adding several negative application conditions to rules is possible. Generally, negative application conditions can be used to restrict unwanted transformations. Nevertheless, there are more concepts for preventing a couple of unwanted transformations. For example, *labeled Petri systems* can be used. Their general idea is quite simple: A label is assigned to every place and a mapping between two places is only allowed if they have the same label. Of course, labeling transitions is also possible, although labeled places are sufficient for most applications. The use of labeled Petri systems combined with negative application conditions is not redundant since labels can only prevent some specific unwanted transformations through the restriction of the mappings and are not able to express conditions like, e.g. restricting the applicability of rules repeatedly.

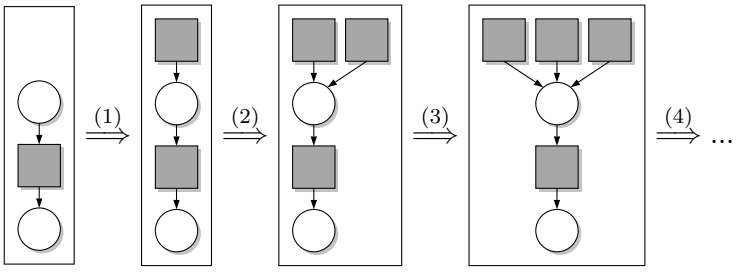


Figure 1.3: Repeated application of rule *addPreTransition*

A lot of theoretical results (for example all of the already mentioned theo-

rems) have to be extended for the use of negative application conditions. In Figure 1.5 it is shown that the basic *Local Church-Rosser Theorem* does not hold in general for the case with negative application conditions since the NAC of rule $addPostTransitionWithNAC$ (see Figure 1.4) forbids the application to the left P/T system. The extension of the most significant results for the use of negative application conditions in general adhesive HLR systems is already introduced by Leen Lambers in [Lam07] and expanded in [LEOP08]. Therefore, the notion of *(weak) adhesive HLR categories with NACs* is introduced. These are (weak) adhesive HLR categories with several additional properties. Additional to the morphism class $\mathcal{M}$ (weak) adhesive HLR categories with NACs have three distinguished morphism classes: $\mathcal{M}'$, $\mathcal{Q}$ and $\mathcal{E}'$ with special properties. $\mathcal{Q}$-morphisms play a very important role for adhesive HLR systems with NACs since they influence the satisfaction of a negative application condition directly. In order to reduce the number of required negative application conditions it is important to choose the morphism class $\mathcal{Q}$ as general as possible. Morphism classes $\mathcal{E}'$ and $\mathcal{M}'$ are already introduced in [EEPT06] for *$\mathcal{E}'$-$\mathcal{M}'$ pair factorization*, *$\mathcal{M}$-$\mathcal{M}'$pushout-pullback decomposition property* and *initial pushouts over $\mathcal{M}'$-morphisms*. These properties are required for some of the most significant results for adhesive HLR systems. They are three of eleven existing additional properties of a (weak) adhesive HLR category with NACs. Furthermore, there are properties like *unique epi-$\mathcal{M}$ factorization*, special pushout-pullback properties and several composition and decomposition properties of special morphisms. The relevance of these properties is described in detail in [Lam07] and [LEOP08].

The main focus of this thesis is to prove that Petri systems (P/T systems, AHL systems as well as labeled P/T systems) fulfill these mentioned properties. Therefore, the categories **PTNet** of P/T nets, **PTSys** of P/T systems, **PTSys**(**L**) of labeled P/T systems, **AHLNet** of AHL nets and **AHLSystems** of AHL systems are introduced in detail and it is proven that these categories are weak adhesive HLR categories with NACs. Additionally, two case studies are given to show a concrete application of the abstract results. The first case study contains the description and the modeling of an airport control system, called *ACS*, with a reconfigurable P/T system. ACS manages the coordination of the airplanes at the airport and ensures that the runways can only be used exclusively by one airplane. ACS can adapt to various changes of the airport. For example, starting and landing runways can be added or removed under several conditions by transforming the basic ACS. The second case study extends ACS to a reconfigurable AHL system, called *AHL-ACS*, with additional functionality, for example managing the airplanes at the gates. Finally, the theoretical results of adhesive HLR systems with NACs are applied to ACS and AHL-ACS.

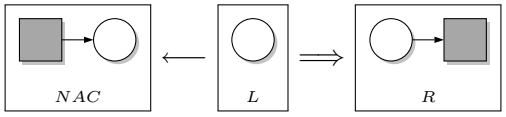


Figure 1.4: Rule $addPostTransitionWithNAC$

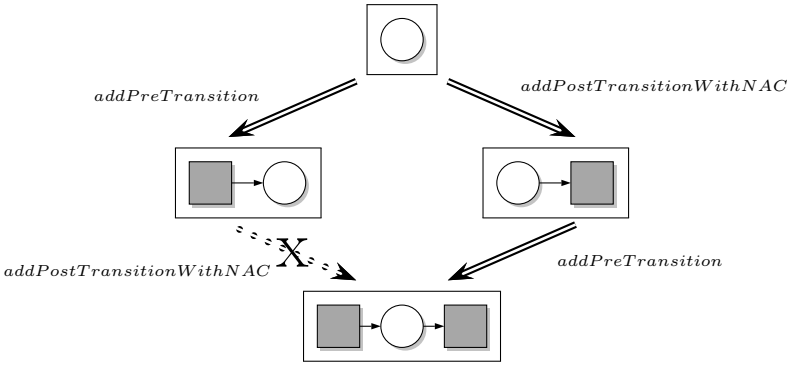


Figure 1.5: Violation of Local Church-Rosser Property

1.2 Structure of this Thesis

This work is structured into four main topics: Adhesive HLR systems with NACs, reconfigurable P/T systems, reconfigurable AHL systems and reconfigurable labeled P/T systems. Chapter 2 contains an introduction to adhesive HLR systems with NACs, where all required properties are defined formally. Additionally, an overview of the most significant notions of transformation systems and the most significant results for adhesive HLR systems with NACs are presented. The content of this chapter is very abstract and serves to show the theoretical background of adhesive HLR systems with NACs. In Chapter 3 reconfigurable P/T systems are analyzed. The first part of this chapter contains a short review of them. Therefore, P/T nets and the category of P/T nets are presented first. Then the extension of P/T nets to P/T systems is performed and the category of P/T systems is introduced. Additional information about these categories can be found in Appendix A with detailed proofs for special morphisms and some categorical constructions. In the next step, it is proven that the category of P/T systems is a weak adhesive category with NACs. Finally, a case study for a reconfigurable P/T system is presented in the case of an airport control system called *ACS*. Additionally, the theoretical results of Chapter 2 are applied to this example. In Chapter 4, P/T nets are extended by data types to AHL nets and the categories of AHL nets and AHL systems are introduced. More detailed information about these categories can be found in Appendix B with detailed proofs for special morphisms and some categorical constructions. Subsequently, it is proven that the category of AHL nets and the category of AHL systems are weak adhesive HLR categories with NACs. Finally, a case study for a reconfigurable AHL system is presented, which is the extension of ACS to AHL systems with additional

functionality. In Chapter 5 labeled P/T systems are introduced. The organization of this chapter is analogous to Chapter 3. In the first step, a review of reconfigurable labeled P/T systems is given. More detailed information about the category of labeled P/T systems can be found in Appendix C with detailed proofs for special morphisms and some categorical constructions. Then it is proven that the category of labeled P/T systems is a weak adhesive HLR category with NACs and, finally, ACS is extended to labeled P/T systems. Additionally, the advantages of labels are identified. Chapter 6 contains the conclusion and a description of future work. Moreover, in Appendix D several detailed proofs can be found. Furthermore, Appendix A contains additional information about the category of sets and Appendix B contains additional information about the categories of algebraic signatures and the category of algebraic specifications.

1.3 Related Work

Graph transformation systems with the double-pushout (DPO) approach and its generalization to adhesive HLR systems are introduced in detail in [EEPT06] and [EPPH06]. [Roz97] compares the double-pushout approach and the single-pushout (SPO) approach. Application conditions were first considered in the 1980s in [EH86], and negative application conditions (NACs) in the 1990s in [HW95] and [HHT96]. For using negative application conditions without losing important qualities of adhesive HLR systems, the extension of (weak) adhesive HLR categories to (weak) adhesive HLR categories with NACs is presented in [LEO06], [Lam07] and [LEOP08] with detailed proofs of the most significant results and theorems. In [PEL07], an approach for the construction of (weak) adhesive HLR categories with some of the required additional properties for the use of negative application conditions is presented. Examples for the use of negative application conditions can be found e.g. in [BTS00], [HHT02], [KMPP05], [MTR05] and [EGdL$^+$05].
P/T nets and P/T systems are introduced in [Rei85] and [NRT92] in original set-theoretical notation. The algebraic notation, basing on a power set or monoid construction, is introduced in [MM90]. [EP04], [HME05] and [EEPT06] contain an introduction to P/T net transformation systems and reconfigurable P/T systems are analyzed in [EEH$^+$07]. A small example of a reconfigurable P/T system with negative application conditions can be found in [RPL$^+$08]. The approach of transforming Petri nets can be restricted to transformations that preserve specific properties as safety or liveness as described in [PU03]. In a series of papers [LO04], [LO06a] and [LO06b], net rewriting systems for describing concurrent systems are introduced. In this context, a system configuration is described as a Petri net and a change of the configuration is described as a graph rewriting rule. This means, rewriting of Petri nets in terms of graph grammars is used for the reconfiguration of nets. These so called marked-controlled reconfigurable nets (MCRN) are extended by control techniques that allow changes of the net for specific markings. The enabling of a rule is not only dependent on the net topology, but also on the marking of specific

control places. *MCReNet* (see [LO06a]) is the corresponding tool for modeling and verification of MCRNs.
For the step from low-level Petri nets to high-level Petri nets algebraic specifications and algebras as introduced in [EM85] and [EM90] are required. In [Pra07] and [PER95] reconfigurable AHL nets are presented and in [Pra08] they are extended to reconfigurable AHL systems by adding markings. Examples of algebraic high-level net transformation systems can be found e.g. in [PER95], [EKMR99] and [Erm96].

Chapter 2

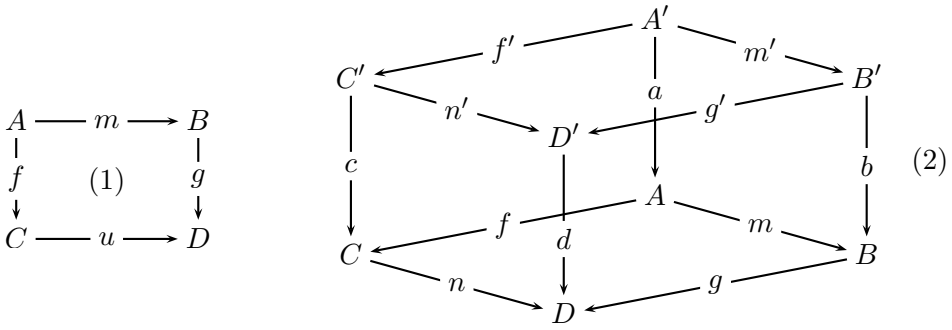
Adhesive HLR Systems with NACs

2.1 Review of Adhesive HLR Categories with NACs

This chapter contains a review of adhesive HLR systems with negative application conditions. In the first part, (weak) adhesive HLR categories are introduced. Subsequently, the additional requirements of adhesive HLR systems with NACs are presented. A more detailed introduction to adhesive HLR systems can be found in [EEPT06]. [Lam07] and [LEOP08] contain detailed information about the expansion of adhesive HLR systems by negative application conditions and [LEO06] introduces conflict detection for graph transformation systems with negative application conditions.

The intuitive idea of (weak) adhesive HLR categories is that of categories with special properties for pushouts and pullbacks. Their formal definition is based on so called *van Kampen squares*. These are commutative squares consisting of pushouts and pullbacks with some special properties as defined in the next definition.

Definition 2.1 (van Kampen Square)**.** A pushout (1) is a van Kampen square if, for any commutative cube (2) with (1) in the bottom and where the back faces are pullbacks, the following statement holds: the top face is a pushout if and only if the front faces are pullbacks.



(2)

Based on this definition of van Kampen squares it is possible to define *adhesive HLR categories*.

Definition 2.2 (Adhesive HLR Category)**.** A category $\mathbf{C}$ with a morphism class $\mathcal{M}$ is called adhesive HLR category if

1. $\mathcal{M}$ is a class of monomorphisms closed under isomorphisms, composition and decomposition,
2. $\mathbf{C}$ has pushouts and pullbacks along $\mathcal{M}$-morphisms and $\mathcal{M}$-morphisms are closed under pushouts and pullbacks,
3. pushouts in $\mathbf{C}$ along $\mathcal{M}$-morphisms are van Kampen squares.

The category $(\mathbf{PTNet}, \mathcal{M})$ of P/T nets and P/T net morphisms with the class $\mathcal{M}$ of all monomorphisms fails to be an adhesive HLR category (see Example 4.23 in [EEPT06]) since the van Kampen square property does not hold. Nevertheless, $(\mathbf{PTNet}, \mathcal{M})$ is a *weak adhesive HLR category*. This is a adhesive HLR category with a weakened van Kampen property. For defining transformation systems, the properties of weak adhesive HLR categories, formalized in the next definition, are sufficient. In the following, the phrase *(weak) adhesive HLR category* means that either an adhesive HLR category or a weak adhesive HLR category can be considered.

Definition 2.3 (Weak Adhesive HLR Category)**.** A category $\mathbf{C}$ with a morphism class $\mathcal{M}$ is called weak adhesive HLR category if

1. $\mathcal{M}$ is a class of monomorphisms closed under isomorphisms, composition and decomposition,
2. $\mathbf{C}$ has pushouts and pullbacks along $\mathcal{M}$-morphisms and $\mathcal{M}$-morphisms are closed under pushouts and pullbacks,
3. pushouts in $\mathbf{C}$ along $\mathcal{M}$-morphisms are weak van Kampen squares, i.e. the VK square property holds for all commutative cubes with $m \in \mathcal{M}$ and ($f \in \mathcal{M}$ or $b, c, d \in \mathcal{M}$) (see Definition 2.1).

As already mentioned, (weak) adhesive HLR categories with NACs contain some additional conditions compared to the case without NACs. These conditions are necessary for using negative application conditions within a transformation system while preserving important qualities like *Local Church-Rosser Theorem*, *Parallelism Theorem*, *Completeness Theorem of Critical Pairs*, *Concurrency Theorem*, *Embedding* and *Extension Theorem* and *Local Confluence Theorem*. All these theorems are presented in the next section of this thesis as well as in [Lam07] and [LEOP08], where they are introduced and proven.

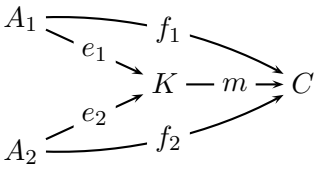
Definition 2.4 ((Weak) Adhesive HLR Category with NACs)**.** A (weak) adhesive HLR category with NACs is a (weak) adhesive HLR category $\mathbf{C}$ with special morphism class $\mathcal{M}$ and in addition three morphism classes $\mathcal{M}'$, $\mathcal{E}'$ and $\mathcal{Q}$ with the following properties:

- unique epi-$\mathcal{M}$ factorization (see Definition 2.5)
- unique $\mathcal{E}'$-$\mathcal{M}'$ pair factorization (see Definition 2.6)
- $\mathcal{M}$-$\mathcal{M}'$ pushout-pullback decomposition property (see Definition 2.7)
- $\mathcal{M}$-$\mathcal{Q}$ pushout-pullback decomposition property (see Definition 2.8)
- initial pushouts over $\mathcal{M}'$-morphisms (see Definition 2.10)
- $\mathcal{M}'$ is closed under pushouts and pullbacks along $\mathcal{M}$-morphisms (see Definition 2.11)
- $\mathcal{Q}$ is closed under pushouts and pullbacks along $\mathcal{M}$-morphisms (see Definition 2.12)
- induced pullback-pushout property for $\mathcal{M}$ and $\mathcal{Q}$ (see Definition 2.13)
- composition property for morphisms in $\mathcal{M}'$ and $\mathcal{Q}$ (see Definition 2.14)
- decomposition property for morphisms in $\mathcal{M}'$ and $\mathcal{Q}$ (see Definition 2.15)
- $\mathcal{Q}$ is closed under composition and decomposition (see Definition 2.16)

The explicit definitions of the itemized properties above follow directly.

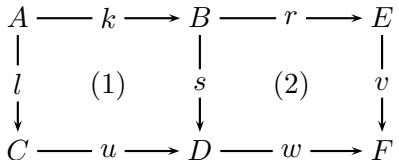
Definition 2.5 (Unique Epi-$\mathcal{M}$ Factorization)**.** A (weak) adhesive HLR category ($\mathbf{C}$,$\mathcal{M}$) has unique epi-$\mathcal{M}$ factorization if, for every morphism $f : A \to C$, there exists an object B, epimorphism $e : A \to B$ and monomorphism $m : B \to C \in \mathcal{M}$ with $m \circ e = f$ and the following universal property holds: For all epimorphisms $e' : A \to B'$ and monomorphisms $m' : B' \to C \in \mathcal{M}$ with $m' \circ e' = f$ exists a unique isomorphism $i : B \to B'$ with $i \circ e = e'$ and $m' \circ i = m$.

Definition 2.6 (Unique $\mathcal{E}'$-$\mathcal{M}'$ Pair Factorization)**.** Given a class of morphism pairs $\mathcal{E}'$ with the same codomain and a morphism class $\mathcal{M}'$, a (weak) adhesive HLR category has $\mathcal{E}'$-$\mathcal{M}'$ pair factorization if, for each pair of morphisms $f_1 : A_1 \to C$ and $f_2 : A_2 \to C$, there exist an object K and morphisms $e_1 : A_1 \to K$, $e_2 : A_2 \to K$ and $m : K \to C$ with $(e_1, e_2) \in \mathcal{E}'$ and $m \in \mathcal{M}'$ such that $m \circ e_1 = f_1$ and $m \circ e_2 = f_2$



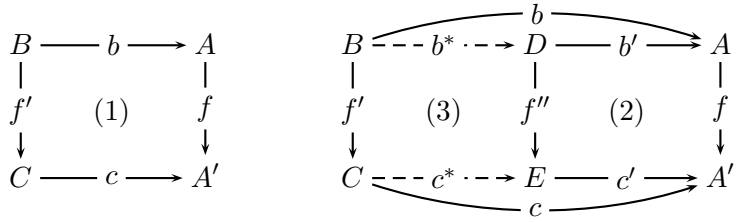
and the following universal property holds:
For every object K' and morphisms $e'_1 : A_1 \to K'$, $e'_2 : A_2 \to K'$ and $m' \in \mathcal{M}'$: $K' \to C$ with $(e'_1, e'_2) \in \mathcal{E}'$, $m' \circ e'_1 = f_1$ and $m' \circ e'_2 = f_2$ there exists a unique isomorphism $i : K \to K'$ with $i \circ e_1 = e'_1$, $i \circ e_2 = e'_2$ and $m' \circ i = m$.

Definition 2.7 ($\mathcal{M}$-$\mathcal{M}'$ Pushout-Pullback Decomposition Property)**.** A (weak) adhesive HLR category ($\mathbf{C}$,$\mathcal{M}$) with a morphism class $\mathcal{M}'$ has the $\mathcal{M}$-$\mathcal{M}'$ pushout-pullback decomposition property if the following property holds:
Given the following commutative diagram with $l \in \mathcal{M}$ and $w \in \mathcal{M}'$, and where (1+2) is a pushout and (2) a pullback, then (1) and (2) are pushouts and also pullbacks:



Definition 2.8 ($\mathcal{M}$-$\mathcal{Q}$ Pushout-Pullback Decomposition Property)**.** Replace morphism class $\mathcal{M}'$ in Definition 2.7 by $\mathcal{Q}$.

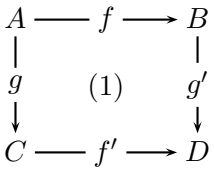
Definition 2.9 (Initial Pushout)**.** Given a morphism $f : A \to A'$ in a (weak) adhesive HLR category, a morphism $b : B \to A$ with $b \in \mathcal{M}$ is called the boundary over f if there is a pushout complement of f and b such that (1) is a pushout which is initial over f. Initiality of (1) over f means, that for every pushout (2) with $b' \in \mathcal{M}$ there exist unique morphisms $b^* : B \to D$ and $c^* : C \to E$ with $b^*, c^* \in \mathcal{M}$ such that $b' \circ b^* = b$, $c' \circ c^* = c$ and (3) is a pushout. B is then called the boundary object and C the context with respect to f.



Definition 2.10 (Initial Pushouts over $\mathcal{M}'$-Morphisms)**.** A (weak) adhesive HLR category with a morphism class $\mathcal{M}'$ has initial pushouts over $\mathcal{M}'$-morphisms if, for every morphism $f : A \to A' \in \mathcal{M}'$, the diagram (see Definition 2.9) can be constructed such that (1) is an initial pushout and $b : B \to A \in \mathcal{M}$.

Definition 2.11 ($\mathcal{M}'$ is closed under Pushouts and Pullbacks along $\mathcal{M}$-Morphisms)**.** Given a (weak) adhesive HLR category (**C**,$\mathcal{M}$) with a morphism class $\mathcal{M}'$, the following statements hold:

1. $\mathcal{M}'$ is closed under pushouts along $\mathcal{M}$-morphisms if for every pushout (1) where $g, g' \in \mathcal{M}$ and $f \in \mathcal{M}'$, it holds that $f' \in \mathcal{M}'$.
2. $\mathcal{M}'$ is closed under pullbacks along $\mathcal{M}$-morphisms if for every pullback (1) where $g, g' \in \mathcal{M}$ and $f' \in \mathcal{M}'$, it holds that $f \in \mathcal{M}'$.



Definition 2.12 ($\mathcal{Q}$ is closed under Pushouts and Pullbacks along $\mathcal{M}$-Morphisms)**.** Replace morphism class $\mathcal{M}'$ in Definition 2.11 by $\mathcal{Q}$.

Definition 2.13 (Induced Pullback-Pushout Property for $\mathcal{M}$ and $\mathcal{Q}$)**.** A (weak) adhesive HLR category (**C**,$\mathcal{M}$) with a morphism class $\mathcal{Q}$ has the induced pullback-pushout property for $\mathcal{M}$ and $\mathcal{Q}$ if the following property holds:
Given morphisms $a : A \to C \in \mathcal{Q}$ and $b : B \to C \in \mathcal{M}$ and the following pullback and pushout,

then the induced morphism $x : E \to C$ with $x \circ e_1 = b$ and $x \circ e_2 = a$ is a monomorphism in $\mathcal{Q}$.

Definition 2.14 (Composition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$)**.** A (weak) adhesive HLR category ($\mathbf{C}$,$\mathcal{M}$) with morphism classes $\mathcal{M}'$ and $\mathcal{Q}$ has the composition property for morphisms in $\mathcal{M}'$ and $\mathcal{Q}$ if the following statement holds:
If $f : A \to B \in \mathcal{Q}$ and $g : B \to C \in \mathcal{M}'$ then $g \circ f \in \mathcal{Q}$.

Definition 2.15 (Decomposition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$)**.** A (weak) adhesive HLR category ($\mathbf{C}$,$\mathcal{M}$) with morphism classes $\mathcal{M}'$ and $\mathcal{Q}$ has the decomposition property for morphisms in $\mathcal{M}'$ and $\mathcal{Q}$ if the following statement holds:
If $g \circ f \in \mathcal{Q}$ and $g \in \mathcal{M}'$ then $f \in \mathcal{Q}$.

Definition 2.16 ($\mathcal{Q}$ is closed under Composition and Decomposition)**.** Given a (weak) adhesive HLR category ($\mathbf{C}$,$\mathcal{M}$) with a morphism class $\mathcal{Q}$.

- $\mathcal{Q}$ is closed under composition if for every pair $f : A \to B \in \mathcal{Q}$, $g : B \to C \in \mathcal{Q}$ of morphisms it holds that $g \circ f \in \mathcal{Q}$.
- $\mathcal{Q}$ is closed under decomposition if for every morphism $g \circ f \in \mathcal{Q}$ with $g \in \mathcal{Q}$ it holds that $f \in \mathcal{Q}$.

2.2 Most Significant Notions of Transformation Systems with NACs

In the next part, the most significant notions of transformation systems are presented. Note that this thesis only summarizes some required basic notions. [EEPT06] contains a detailed introduction to transformation systems.

Transformation systems presented in this thesis are rule-based, i.e. transformation steps are described by rules as formalized in the following definition.

Definition 2.17 (Rule / Production)**.** A rule in a (weak) adhesive HLR category ($\mathbf{C}$,$\mathcal{M}$) is given by three $\mathbf{C}$-objects L, K and R and two $\mathbf{C}$-morphisms $l : K \to L \in \mathcal{M}$ and $r : K \to R \in \mathcal{M}$. [1]

According to the DPO approach presented in [EEPT06], the direct transformation of an object via the rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ is described by a double-pushout diagram along the morphisms l and r of the rule. The formal definition follows later.

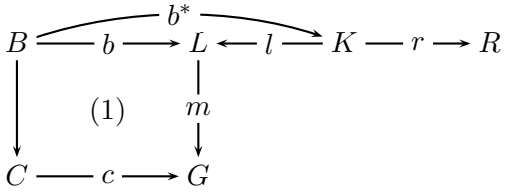
$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & K & \xrightarrow{\ r\ } & R \\ \downarrow{\scriptstyle m} & (1) & \downarrow{\scriptstyle k} & (2) & \downarrow{\scriptstyle n} \\ G & \xleftarrow{\ f\ } & D & \xrightarrow{\ d\ } & H \end{array}$$

Since the morphisms of a rule are $\mathcal{M}$-morphisms, the existence of pushouts along these morphisms is guaranteed. Although, for a given morphism $m : L \to G$, called

[1] A rule is also called production.

match, from the left-hand side of the rule L to an object G, the existence of the so called *pushout complement* D is not ensured. Therefore, applying a rule to an object is not always possible and a necessary and sufficient condition for the applicability of rules is required. This so called *gluing condition* is formalized in the next definition.

Definition 2.18 (Gluing Condition)**.** Given an adhesive HLR system AHS over a (weak) adhesive HLR category with initial pushouts, then a match $m : L \to G$ satisfies the gluing condition with respect to a production $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ if, for the initial pushout (1) over m, there is a morphism $b^* : B \to K$ such that $l \circ b^* = b$:



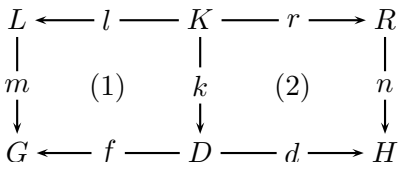
In this case, $b, l \in \mathcal{M}$ implies $b^* \in \mathcal{M}$ by the decomposition property of $\mathcal{M}$.

Remark 2.19. This is the categorical formulation of the gluing condition. Corresponding gluing conditions are formalized for most instances of well known transformation systems, for example graphs, P/T nets and systems and AHL nets and systems. The gluing conditions for P/T systems, AHL nets and AHL systems are defined in the appendix of this thesis (see Facts A.21, B.26 and B.32).

According to the gluing condition, the applicability of rules and transformations are formalized in the following definitions.

Definition 2.20 (Applicability of a Rule)**.** A rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ is applicable for a match $m : L \to G$ if and only if m fulfills the gluing condition. A match m that fulfills the gluing condition is also called consistent with respect to p.

Definition 2.21 (Direct Transformation via a Rule)**.** Given a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ and a consistent match $m : L \to G$ with respect to p. Then the direct transformation is given by the following diagram, where (1) and (2) are pushouts:



H is called the result of the direct transformation. A sequence $G_0 \Rightarrow G_1 \Rightarrow ... \Rightarrow G_n$ of direct transformations is called a transformation and is denoted $G_0 \overset{*}{\Rightarrow} G_n$. G_n is called the result of the transformation.

Now, the theory is extended by *negative application conditions*. Negative application conditions afford the possibility to restrict the application of a rule if a certain structure is present. Analogous to negative application conditions, there are *application conditions* (see Definition 7.6 in [EEPT06]). Although, only negative application conditions are considered in this thesis since they are the mostly used kind of application conditions.
As already mentioned in the Inroduction, there are *left negative application conditions*, which forbid the existence of a certain structure before applying a rule, and there are *right negative application conditions*, which forbid the existence of a certain structure after applying a rule. In this thesis, only *left negative application conditions* are considered and they are simply called *negative application conditions* or shortly *NACs*.

Definition 2.22 ((Left) Negative Application Condition)**.** A simple negative application condition of a production $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ in a (weak) adhesive HLR category with NACs $(C, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is of the form $NAC(n)$, where $n : L \to N$ is a morphism.
A morphism $m : L \to G$ satisfies $NAC(n)$, written $m \models NAC(n)$, if there does not exist a morphism $q : N \to G \in \mathcal{Q}$ with $q \circ n = m$.

Remark 2.23. In this thesis, only adhesive HLR systems with rules having an empty set of right negative application conditions are considered. This is without loss of generality because each right NAC can be translated into an equivalent left NAC as explained in Definition 2.9 in [Lam07]. See also Lemma 2.11 and Remark 2.8 in [Lam07].

According to the case without negative application conditions, rules with negative application conditions and their applicability are defined in the following.

Definition 2.24 (Rule / Production with NACs)**.** A rule $p = (L \leftarrow K \rightarrow R)$ (see Definition 2.17) in a (weak) adhesive HLR category with NACs $(C, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ with a set of NACs $NAC_p = \{NAC(n_i) | n_i : L \to N_i,\ i \in I\}$ (see Definition 2.22) is called rule with NACs, where I is an index set.

Definition 2.25 (Applicability of a Rule with NACs)**.** Given a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ with a set of negative application conditions NAC_p and a consistent match $m : L \to G$ with respect to p. Then the rule p is applicable if and only if m satisfies all NACs of the set NAC_p.

Finally, the definition of an adhesive HLR system with NACs is given.

Definition 2.26 (Adhesive HLR System with NACs)**.** An adhesive HLR system with NACs $AHS = (C, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}, P)$ consists of a (weak) adhesive HLR category with NACs $(C, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ as formalized in Definition 2.4 and a set of rules with NACs P (see Definition 2.24).

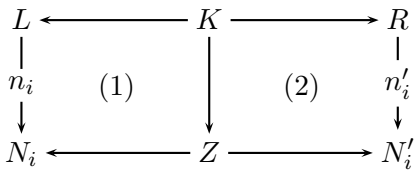
2.3 Most Significant Results for Adhesive HLR Systems with NACs

This chapter contains the most important theorems for adhesive HLR systems with NACs and additional required definitions. A more detailed description, small examples and detailed proofs of the following theorems can be found in [Lam07] and [LEOP08].

2.3.1 Inverse Rules

A requirement for a lot of transformation systems is the possibility to revoke a transformation step. In the case without negative application conditions, it is quite obvious that for every given rule an inverse rule exists. In the case with negative application conditions, this requirement demands the possibility to translate right NACs into left NACs. As already mentioned, this possibility is given in the case of adhesive HLR systems with NACs. The Theorem *Inverse Direct Transformation with NACs*, given in this subsection, formalizes this result.

Definition 2.27 (Construction of NACs on Inverse Rule)**.** For each $NAC(n_i)$ with $n_i : L \to N_i$ on $p = (L \leftarrow K \rightarrow R)$, the equivalent NAC $R_p(NAC(n_i))$ on the inverse rule $p^{-1} = (R \leftarrow K \rightarrow L)$ is defined in the following way:



- If the pair $(K \to L, L \to N_i)$ has a pushout complement, construct $(K \to Z, Z \to N_i)$ as pushout complement (1). Then construct pushout (2) with the morphism $n_i' : R \to N_i'$ and define $R_p(NAC(n_i)) = NAC(n_i')$.
- If the pair $(K \to L, L \to N_i)$ does not have a pushout complement, define $R_p(NAC(n_i)) = true$.

For each set of NACs on p $NAC_p = \bigcup_{i \in I} NAC(n_i)$, define the following set of NACs on p^{-1} : $NAC_{p^{-1}} = R_p(NAC_p) = \bigcup_{i \in I'} R_p(NAC(n_i))$ with $i \in I'$ if and only if the pair $(K \to L, L \to N_i)$ has a pushout complement.

Remark 2.28 (Translation of NACs). In [Lam07] and [LEOP08] a leftward translation $L_p(NAC_p)$ of a set of right NACs NAC_p into a set of left NACs is introduced. It is used to create the NACs of the inverse production. This approach is equivalent to the construction of NACs on inverse rules presented in Definition 2.27.

Theorem 2.29 (Inverse Direct Transformation with NACs)**.** *For each direct transformation with NACs $G \Rightarrow H$ via a rule $p : L \leftarrow K \rightarrow R$ with NAC_p a set of NACs on p, there exists an inverse direct transformation with NACs $H \Rightarrow G$ via the inverse rule p^{-1} with $NAC_{p^{-1}}$.*

2.3.2 Parallelism

Parallelism is an important concept for every transformation system. Therefore, the notion of independent direct transformations is necessary. Intuitively, it is quite obvious that the order of some special (i.e. *independent*) transformations does not matter. For example, in this context it makes no difference for the construction of an airport if the starting runway or the landing runway is built first. Additionally, there is the possibility to build both runways simultaneously. This also leads to the same result of an airport with a starting and a landing runway. Especially, because a lot of transformation systems are infinite, it is important to formalize these results as universally valid theorems. The *Local Church-Rosser Theorem with NACs* and the *Parallelism Theorem with NACs*, presented in this subsection, represent the generalization of the mentioned examples. All for these theorems required definitions are also formalized in this subsection.
Note that the *Parallelism Theorem with NACs* requires binary coproducts compatible with $\mathcal{M}$ and an additional composition property for $\mathcal{Q}$-morphisms.

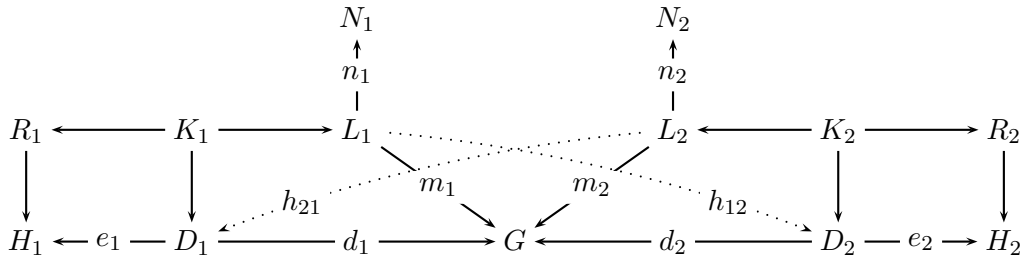
Definition 2.30 (Parallel and Sequential Independence with NACs)**.** Two direct transformations $G \overset{p_1,m_1}{\Rightarrow} H_1$ with NAC_{p_1} and $G \overset{p_2,m_2}{\Rightarrow} H_2$ with NAC_{p_2} are *parallel independent* if

$$\exists h_{12} : L_1 \rightarrow D_2 : (d_2 \circ h_{12} = m_1 \wedge e_2 \circ h_{12} \models NAC_{p_1})$$

and

$$\exists h_{21} : L_2 \rightarrow D_1 : (d_1 \circ h_{21} = m_2 \wedge e_1 \circ h_{21} \models NAC_{p_2})$$

as in the following diagram:



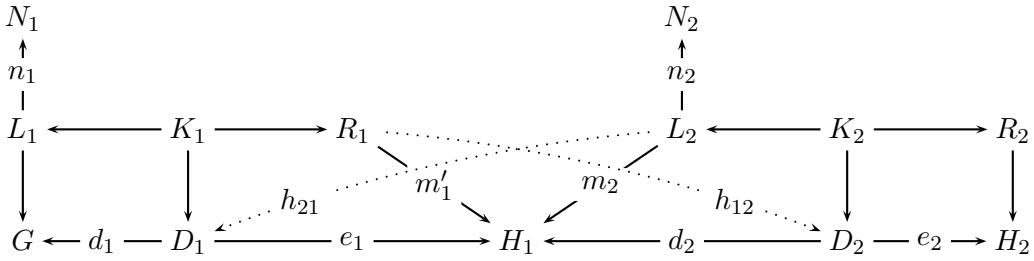
Two direct transformations $G \overset{p_1,m_1}{\Rightarrow} H_1$ with NAC_{p_1} and $G \overset{p_2,m_2}{\Rightarrow} H_2$ with NAC_{p_2} are *sequentially independent* if

$$\exists h_{12} : R_1 \rightarrow D_2 : (d_2 \circ h_{12} = m_1' \wedge e_2 \circ h_{12} \models NAC_{p_1^-1})$$

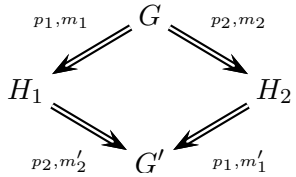
and

$$\exists h_{21} : L_2 \to D_1 : (e_1 \circ h_{21} = m_2 \wedge d_1 \circ h_{21} \models NAC_{p_2})$$

as in the following diagram:



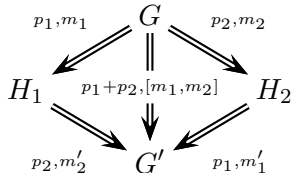
Theorem 2.31 (Local Church-Rosser Theorem with NACs)**.** *Given an adhesive HLR system with NACs AHS and two parallel independent (see Definition 2.30) direct transformations with NACs* $H_1 \overset{p_1,m_1}{\Leftarrow} G \overset{p_2,m_2}{\Rightarrow} H_2$*, there are an object* G' *and direct transformations* $H_1 \overset{p_2,m_2'}{\Rightarrow} G'$ *and* $H_2 \overset{p_1,m_1'}{\Rightarrow} G'$ *such that* $G \overset{p_1,m_1}{\Rightarrow} H_1 \overset{p_2,m_2'}{\Rightarrow} G'$ *and* $G \overset{p_2,m_2}{\Rightarrow} H_2 \overset{p_1,m_1'}{\Rightarrow} G'$ *are sequentially independent. Vice versa, given two sequentially independent (see Definition 2.30) direct transformations with NACs* $G \overset{p_1,m_1}{\Rightarrow} H_1 \overset{p_2,m_2'}{\Rightarrow} G'$ *there are an object* H_2 *and sequentially independent direct transformations* $G \overset{p_2,m_2}{\Rightarrow} H_2 \overset{p_1,m_1'}{\Rightarrow} G'$ *such that* $H_1 \overset{p_1,m_1}{\Leftarrow} G \overset{p_2,m_2}{\Rightarrow} H_2$ *are parallel independent:*



Definition 2.32 (Parallel Rule and Transformation with NACs)**.** Let $AHS = (\mathbf{C}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}, P)$ be an adhesive HLR system with NACs, where $(\mathbf{C}, \mathcal{M})$ has binary coproducts compatible with $\mathcal{M}$ (see Definition 5.14 in [EEPT06]). Given two rules $p_i = L_i \overset{l_i}{\leftarrow} K_i \overset{r_i}{\rightarrow} R_i$ with NAC_{p_i} for $i = 1, 2$, the *parallel rule* $p_1 + p_2$ with $NAC_{p_1+p_2}$ is defined by the coproduct constructions over the corresponding objects and morphisms: $p_1 + p_2 = L_1 + L_2 \overset{l_1+l_2}{\leftarrow} K_1 + K_2 \overset{r_1+r_2}{\rightarrow} R_2 + R_2$ and $NAC_{p_1+p_2} = \{n_1 + id_{L_2} | n_1 \in NAC_{p_1}\} \cup \{id_{L_1} + n_2 | n_2 \in NAC_{p_2}\}$.
A direct transformation $G \Rightarrow G'$ via $p_1 + p_2$ with $NAC_{p_1+p_2}$ and a match $m : L_1 + L_2 \to G$ satisfying $NAC_{p_1+p_2}$ is a *direct parallel transformation with NACs* or *parallel transformation with NACs* for short.

Theorem 2.33 (Parallelism Theorem with NACs: Synthesis)**.** *Let* $AHS = (\mathbf{C}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}, P)$ *be an adhesive HLR system with NACs, where* $(\mathbf{C}, \mathcal{M})$ *has binary coproducts compatible with* $\mathcal{M}$ *and where the composition of a coproduct morphism with a morphism in* $\mathcal{Q}$ *is again in* $\mathcal{Q}$*.*

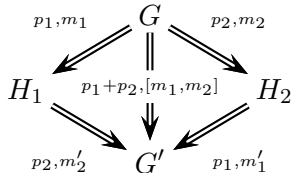
Then given a sequentially independent (see Definition 2.30) direct transformation sequence with NACs $G \Rightarrow H_1 \Rightarrow G'$ *via* p_1, m_1 *(resp.* p_2, m_2'*) with* NAC_{p_1} *(resp.* NAC_{p_2}*), there is a construction leading to a parallel transformation with NACs* $G \Rightarrow G'$ *via* $[m_1, m_2]$ *and the parallel rule* $p_1 + p_2$ *with* $NAC_{p_1+p_2}$*, called a* synthesis construction*.*



Definition 2.34 (NAC-Compatible Parallel Transformation)**.** Given a parallel transformation with NACs $G \Rightarrow G'$ via match $m : L_1 + L_2 \to G$ and the parallel rule $p_1 + p_2$ with $NAC_{p_1+p_2}$. Let $m_1 : L_1 \to G$, $m_2 : L_2 \to G$ be the matches of the direct transformations $G \Rightarrow H_1$ and $G \Rightarrow H_2$ via p_1 resp. p_2 and m_1' and m_2' the matches of the direct transformations $H_2 \Rightarrow G'$ and $H_1 \Rightarrow G'$ via p_1 resp. p_2 as constructed in the Parallelism Theorem without NACs (Analysis part in Theorem 5.18 in [EEPT06]). The parallel transformation with NACs $G \Rightarrow G'$ is *NAC-compatible* if $m_1, m_1' \models NAC_{p_1}$ and $m_2, m_2' \models NAC_{p_2}$.

Theorem 2.35 (Parallelism Theorem with NACs: Analysis)**.** *Let* $AHS = (\mathbf{C}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}, P)$ *be an adhesive HLR system with NACs, where* $(\mathbf{C}, \mathcal{M})$ *has binary coproducts compatible with* $\mathcal{M}$ *and where the composition of a coproduct morphism with a morphism in* $\mathcal{Q}$ *is again in* $\mathcal{Q}$*.*

- *Given a* NAC-compatible *direct parallel transformation with NACs* $G \Rightarrow G'$ *via* $m : L_1 + L_2 \to G$ *and the parallel rule* $p_1 + p_2$ *with* $NAC_{p_1+p_2}$*, then there is a construction leading to two sequentially independent (see Definition 2.30) transformation sequences with NACs* $G \Rightarrow H_1 \Rightarrow G'$ *via* p_1, m_1 *and* p_2, m_2' *and* $G \Rightarrow H_2 \Rightarrow G'$ *via* p_2, m_2 *and* p, m_1'*, called an* analysis construction*.*
- Bijective Correspondence. *The synthesis construction of Theorem 2.33 and the analysis construction are inverse to each other up to isomorphism.*



2.3.3 Critical Pairs

A critical pair describes a conflict between two transformations in a minimal context. The morphism class $\mathcal{E}'$ is required to express this minimal context. The critical pairs and their completeness are a significant concept because an infinite number of

transformations, which are in conflict, can be reduced to a finite number of critical pairs (under the assumption that the set of rules is finite). Critical pairs are used to show the (local) confluence of a transformation system. Local confluence means, that for every given pair of tranformations of one object, transformation sequences of both results exist, such that the final result is the same. A formal definition of local confluence follows later.
In this subsection, the *Completeness Theorem of Critical Pairs with NACs*, expressing the possibility to embed a critical pair into a pair of transformations in conflict, is presented. Therefore, it is necessary to define conflicts of direct transformations with NACs, critical pairs themselves and extension diagrams first.
Note that $\mathcal{E}'$-$\mathcal{M}'$ pair factorization (see Definition 2.6) as well as $\mathcal{M}$-$\mathcal{M}'$ pushout-pullback decomposition property (see Definition 2.7) are required even in the case without NACs for the *completeness of critical pairs* (see Lemma 6.22 in [EEPT06]).

Definition 2.36 (Conflict of Direct Transformations with NACs)**.** Two direct transformations $G \overset{p_1,m_1}{\Rightarrow} H_1$ with NAC_{p_1} and $G \overset{p_2,m_2}{\Rightarrow} H_2$ with NAC_{p_2} are in conflict if and only if they are not parallel independent (see Definition 2.30).

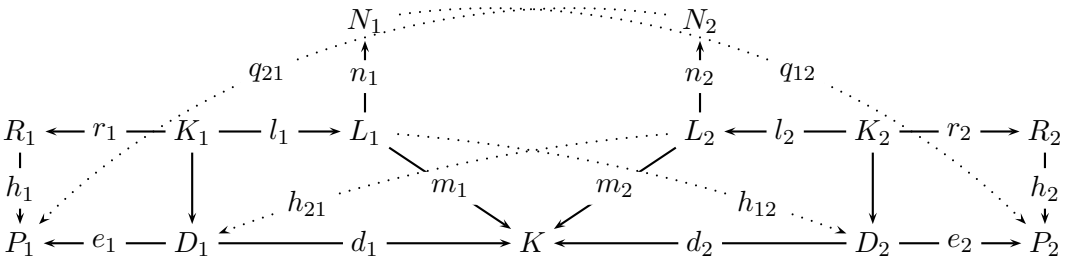
Remark 2.37. A conflict characterization with a detailed proof can be found in Lemma 3.10 in [Lam07].

Definition 2.38 (Critical Pair with NACs)**.** A critical pair is a pair of direct transformations $K \overset{p_1,m_1}{\Rightarrow} P_1$ with NAC_{p_1} and $K \overset{p_2,m_2}{\Rightarrow} P_2$ with NAC_{p_2} such that:

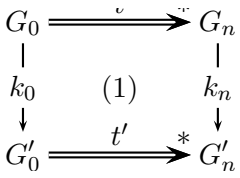
1. (a) $\nexists h_{12} : L_1 \to D_2 : d_2 \circ h_{12} = m_1$ and $(m_1, m_2) \in \mathcal{E}'$. (use-delete-conflict)
   or
   (b) $\exists h_{12} : L_1 \to D_2 : d_2 \circ h_{12} = m_1$, but for one of the NACs $n_1 : L_1 \to N_1$ of p_1 there exists a morphism $q_{12} : N_1 \to P_2 \in \mathcal{Q} : q_{12} \circ n_1 = e_2 \circ h_{12}$ and $(q_{12}, h_2) \in \mathcal{E}'$. (forbid-produce-conflict)

   or

2. (a) $\nexists h_{21} : L_2 \to D_1 : d_1 \circ h_{21} = m_2$ and $(m_1, m_2) \in \mathcal{E}'$. (delete-use-conflict)
   or
   (b) $\exists h_{21} : L_2 \to D_1 : d_1 \circ h_{21} = m_2$, but for one of the NACs $n_2 : L_2 \to N_2$ of p_2 there exists a morphism $q_{21} : N_2 \to P_1 \in \mathcal{Q} : q_{21} \circ n_1 = e_1 \circ h_{21}$ and $(q_{21}, h_1) \in \mathcal{E}'$. (produce-forbid-conflict)

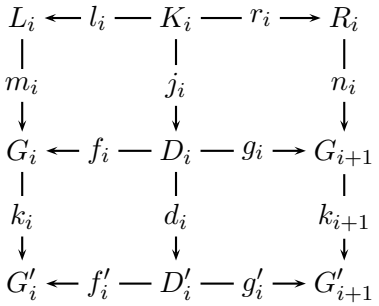


Definition 2.39 (Extension Diagram with NACs)**.** An extension diagram is a diagram (1),

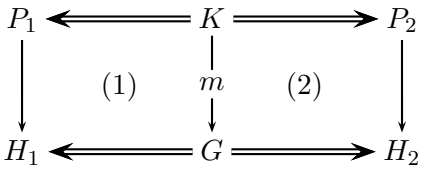


where $k_0 : G_0 \to G'_0$ is a morphism, called extension morphism, and $t : G_0 \overset{*}{\Rightarrow} G_n$ and $t' : G'_0 \overset{*}{\Rightarrow} G'_n$ are transformations with NACs via the same rules $(p_0, ..., p_{n-1})$ and matches $(m_0, ..., m_{n-1})$ and extended matches $(k_0 \circ m_0, ..., k_{n-1} \circ m_{n-1})$ respectively, defined by the following DPO diagrams:

p_i :



Theorem 2.40 (Completeness of Critical Pairs with NACs)**.** *For each pair of direct transformations* $H_1 \overset{p_1,m'_1}{\Leftarrow} G \overset{p_2,m'_2}{\Rightarrow} H_2$ *in conflict there is a critical pair with extension diagrams (1) and (2) and* $m \in \mathcal{M}'$*.*



2.3.4 Concurrency

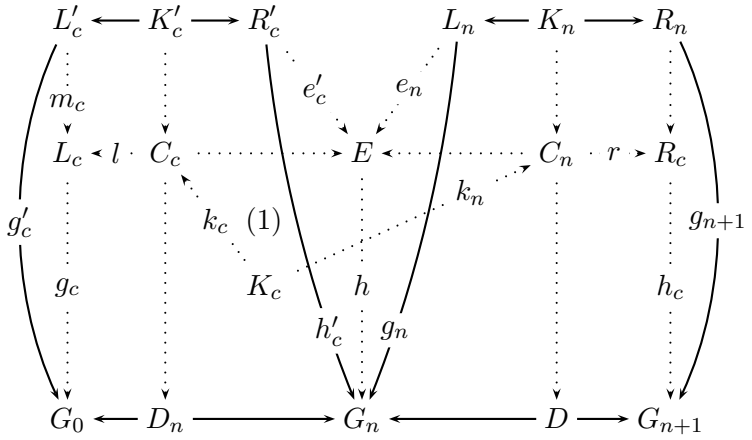
Through the *Parallelism Theorem with NACs* it is possible to summarize several independent direct transformations into one equivalent direct transformation. However, in general there will be dependencies between several direct transformations of a transformation sequence. In this case, it is possible to use the *Concurrency Theorem with NACs*. This allows to translate every transformation sequence into an equivalent direct transformation. Therefore, it is necessary to create the *concurrent rule with NACs*, as formalized in the following definition. The concurrent rule contains a set of NACs consisting all translated NACs occuring in the transformation sequence. This translation of NACs is defined in Section 5 in [Lam07] and Section 4 in [LEOP08]. Certainly, since the *Concurrency Theorem with NACs* is one of

the most significant results for adhesive HLR systems with NACs, it is formulated explicitly in the following part of this thesis.
Note that the construction of E-related transformations as described in Fact 5.29 in [EEPT06] requires $\mathcal{E}'$-$\mathcal{M}'$ pair factorization (see Definition 2.6) as well as $\mathcal{M}$-$\mathcal{M}'$ pushout-pullback decomposition property (see Definition 2.7), even in the case without NACs.

Definition 2.41 (Concurrent Rule with NACs, induced (co-, lhs-) match)**.**

$n = 0$ For a direct transformation $G_0 \Rightarrow G_1$ via match $g_0 : L_0 \to G_0$, comatch $g_1 : R_1 \to G_1$ and rule $p_0 : L_0 \leftarrow K_0 \to R_0$ with NAC_{p_0} the *concurrent rule* p_c *with NACs* induced by $G_0 \Rightarrow G_1$ is defined by $p_c = p_0$ with $NAC_{p_c} = NAC_{p_0}$, the *concurrent comatch* h_c is defined by $h_c = g_1$, the *concurrent lhs-match* by $id : L_0 \to L_0$ and the *concurrent match* g_c by $g_c = g_0 : L_0 \to G_0$.

$n \geq 1$ Consider $p'_c : L'_c \leftarrow K'_c \to R'_c$ (resp. $g'_c : L'_c \to G_0$, $h'_c : R'_c \to G_n$, $m'_c : L_0 \to L'_c$), the concurrent rule with NACs (resp. concurrent match, comatch, lhs-match) induced by $G_0 \overset{n}{\Rightarrow} G_n$. Let $((e'_c, e_n), h)$ be the $\mathcal{E}'$-$\mathcal{M}'$ pair factorization of the comatch h'_c and match g_n or $G_n \Rightarrow G_{n+1}$. According to Fact 5.29 in [EEPT06] pushout-pullback decomposition, pushout composition and decomposition lead to the diagram below in which (1) is a pullback and all other squares are pushouts:



For a transformation sequence $G_0 \overset{n+1}{\Rightarrow} G_{n+1}$ the *concurrent rule* p_c *with NACs* (resp. concurrent match, comatch, lhs-match) induced by $G_0 \overset{n+1}{\Rightarrow} G_{n+1}$ is defined by $p_c = L_c \overset{l \circ k_c}{\leftarrow} K_c \overset{r \circ k_n}{\to} R_c$ ($g_c : L_c \to G_0$, $h_c : R_c \to G_{n+1}$, $m_c \circ m'_c : L_0 \to L_c$). Thereby NAC_{p_c} is defined by $NAC_{p_c} = DL_{p_c}(NAC_{L_n}) \cup D_{m_c}(NAC_{L'_c})$.

Remark 2.42. $D_{m_c}(NAC_{L'_c})$ is the downward translation of $NAC_{L'_c}$ with respect to match m_c (see Definition 5.1, Lemma 5.2, Remark 5.3 and Definition 5.4 in [Lam07]). $DL_{p_c}(NAC_{L_n})$ is the down- and leftward translation of NAC_{L_n} with respect to rule p_c (see Def. 5.6 and Lemma 5.7 in [Lam07])

Theorem 2.43 (Concurrency Theorem with NACs)**.**

(i) Synthesis. *Given a transformation sequence* $t : G_0 \overset{*}{\Rightarrow} G_{n+1}$ *via a sequence of rules* $p_1, p_2, ..., p_n$*, then there is a* synthesis construction *leading to a direct transformation* $G_0 \Rightarrow G_{n+1}$ *via the concurrent rule* $p_c : L_c \leftarrow K_c \rightarrow R_c$ *with* NAC_{p_c}*, match* $g_c : L_c \rightarrow G_0$ *and comatch* $h_c : R_c \rightarrow G_{n+1}$ *induced by* $t : G_0 \overset{*}{\Rightarrow} G_{n+1}$*.*

(ii) Analysis. *Given a direct transformation* $G'_0 \Rightarrow G'_{n+1}$ *via the concurrent rule* $p_c : L_c \leftarrow K_c \rightarrow R_c$ *with* NAC_{p_c} *induced by* $t : G_0 \overset{*}{\Rightarrow} G_{n+1}$ *via a sequence of rules* $p_1, p_2, ..., p_n$*, then there is an* analysis construction *leading to a transformation sequence* $t' : G'_0 \overset{*}{\Rightarrow} G'_{n+1}$ *with NACs via* $p_1, p_2, ..., p_n$*.*

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

2.3.5 Embedding and Extension

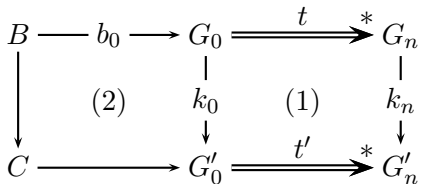
Under several conditions, a transformation $t : G_0 \overset{*}{\Rightarrow} G_n$ can be extended to a transformation $t' : G'_0 \overset{*}{\Rightarrow} G'_n$ via an extension morphism $k_0 : G_0 \rightarrow G'_0$. The transformation t' is based on the same rules as t. The *Embedding Theorem with NACs* describes a condition for the existence of this so called embedding and the *Extension Theorem with NACs* states that this condition is not only sufficient, but also necessary. A vertical composition of extension diagrams under several conditions is possible through these theorems.
The *Embedding Theorem with NACs* requires an additional property compared to the case without NACs named *NAC-consistency*. This property is formalized in the following definition.
Note that the *Embedding* and the *Extension Theorem* require *initial pushouts over* $\mathcal{M}'$*-morphisms* (see Definition 2.10) even in the case without NACs.

Definition 2.44 (NAC-Consistency)**.** A morphism $k_0 : G_0 \rightarrow G'_0$ is called NAC-consistent with respect to a transformation $t : G_0 \overset{*}{\Rightarrow} G_n$ if $k_0 \circ g_c \models NAC_{p_c}$ with NAC_{p_c} the *concurrent NAC* and g_c the *concurrent match induced by* t*.*

Theorem 2.45 (Embedding Theorem with NACs)**.** *Given a transformation* $t : G_0 \overset{n}{\Rightarrow} G_n$ *with NACs. If* $k_0 : G_0 \rightarrow G'_0$ *is boundary consistent (i.e. consistency as in Definition 6.12 in [EEPT06]) and NAC-consistent (see Definition 2.44) with respect to* t *then there exists an extension diagram with NACs over* t *and* k_0*.*

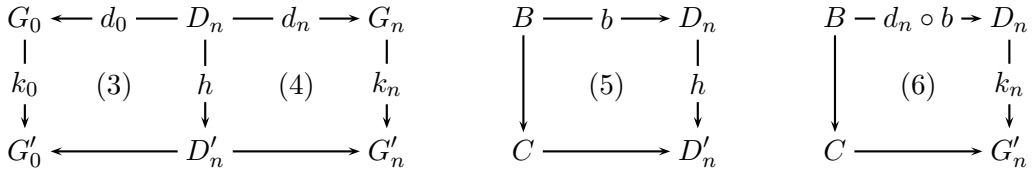
Theorem 2.46 (Extension Theorem with NACs)**.** *Given a transformation* $t : G_0 \overset{n}{\Rightarrow} G_n$ *with NACs with a derived span* $der(t) = (G_0 \overset{d_0}{\leftarrow} D_n \overset{d_n}{\rightarrow} G_n)$ *(see Definition 6.9*

in [EEPT06]) and an extension diagram (1) as in the following picture:



then

- $k_0 : G_0 \to G'_0$ *is boundary consistent with respect to* t*, with the morphism* $b : B \to D_n$*.*
- $k_0 : G_0 \to G'_0$ *is NAC-consistent with respect to* t*.*
- *Let* p_c *(resp.* g_c*) be the concurrent rule with* NAC_{p_c} *(resp. concurrent match) induced by* t*. There is a direct transformation* $G'_0 \Rightarrow G'_n$ *via* $der(t)$ *with* $NAC_{der(t)} = D_{g_c}(NAC_{p_c})$ *and match* k_0 *given by pushouts (3) and (4) with* $h, k_n \in \mathcal{M}'$*.*
- *There are initial pushouts (5) and (6) over* $h \in \mathcal{M}'$ *(see Definition 2.9) and* $k_n \in \mathcal{M}'$*, respectively, with the same boundary-context morphism* $B \to C$*.*

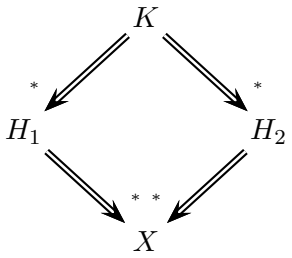


2.3.6 Confluence

Confluence describes the behaviour of a whole transformation system or of a pair of transformations. A pair of transformations of the same source object is called confluent if it is possible to transform the two results of the single transformations into the same object. A transformation system is called locally confluent if this property holds for every pair of transformations. The formal definition of (local) confluence is given in this subsection. Confluence is the main property of interest for adhesive HLR systems (with NACs). The reasons therefore are quite obvious. Every transformation system which provides multiple possibilities to transform the start object into a different object shows this behaviour for at least one pair of transformations. The *Local Confluence Theorem with NACs*, presented in this subsection, expresses a sufficient condition for the local confluence of an adhesive HLR system with NACs. Therefore, a property named *strict NAC-confluence* is required.

Note that the *Local Confluence Theorem* even in the case without NACs (see Definition 6.28 in [EEPT06]) requires *initial pushouts over $\mathcal{M}'$-morphisms* (see Definition 2.10), $\mathcal{E}'$-$\mathcal{M}'$ pair factorization (see Definition 2.6) as well as $\mathcal{M}$-$\mathcal{M}'$ pushout-pullback decomposition property (see Definition 2.7).

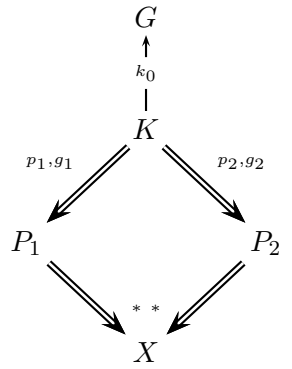
Definition 2.47 ((Local) Confluence)**.** A pair of transformations (with NACs) $H_1 \overset{*}{\Leftarrow} G \overset{*}{\Rightarrow} H_2$ is *confluent* if there are transformations $H_1 \overset{*}{\Rightarrow} X$ and $H_2 \overset{*}{\Rightarrow} X$:



An adhesive HLR system (with NACs) is *locally confluent* if this property holds for each pair of direct transformations. The system is *confluent* if this holds for all pairs of transformations.

Definition 2.48 (Strict NAC-Confluence of Critical Pairs)**.** A critical pair $P_1 \overset{p_1,g_1}{\Leftarrow} K \overset{p_2,g_2}{\Rightarrow} P_2$ is *strictly NAC-confluent* if and only if

- it is *strictly confluent* via some transformations $t_1 : K \overset{p_1,g_1}{\Rightarrow} P_1 \overset{*}{\Rightarrow} X$ and $t_2 : K \overset{p_2,g_2}{\Rightarrow} P_2 \overset{*}{\Rightarrow} X$ (see Definition 6.26 in [EEPT06])
- and it is *NAC-confluent for t_1 and t_2* i.e. for every morphism $k_0 : K \to G \in \mathcal{M}'$ which is NAC-consistent with respect to $K \overset{p_1,g_1}{\Rightarrow} P_1$ and $K \overset{p_2,g_2}{\Rightarrow} P_2$ it follows that k_0 is NAC-consistent with respect to t_1 and t_2.



Theorem 2.49 (Local Confluence Theorem with NACs)**.** *Given an adhesive HLR system with NACs, it is locally confluent, if all its critical pairs are strictly NAC-confluent.*

Chapter 3

Reconfigurable P/T Systems with NACs

3.1 Review of Reconfigurable P/T Systems

This section contains a review of reconfigurable P/T systems with the most significant definitions and facts with respect to this thesis. In the first subsection, P/T nets and the category **PTNet** of P/T nets and P/T net morphisms are introduced. For the sake of completeness, special morphisms, coproducts, pushouts and pullbacks along monomorphisms of this category are presented in Appendix A.1 and the correctness of these constructions is proven.
In the second subsection, P/T nets are expanded to P/T systems by adding markings. The category **PTSys** of P/T systems and morphisms is introduced. Additionally, the most important concepts of this category, with respect to this thesis, are introduced in Appendix A.2 and their correctness is proven.

3.1.1 P/T Nets and the Category of P/T Nets

A P/T net is a bipartite and directed graph, consisting of *places* and *transitions* as nodes. There are several formal definitions of P/T nets. In this thesis, the well-known monoid notation presented in [MM90] is used. The next definitions formalize P/T nets and P/T net morphisms and introduce special P/T net morphisms. Finally, the category **PTNet** of P/T nets and P/T net morphisms is introduced.

Definition 3.1 (P/T Net)**.** A P/T net is given by $PN = (P, T, pre, post)$ with places P, transitions T, and pre- and post-domain functions $pre, post : T \to P^{\oplus}$.

Remark 3.2. $P^{\oplus}$ is the free commutative monoid over P (see Section 2 in [EEH+07]).

Definition 3.3 (P/T Net Morphism (**PTNet**-Morphism))**.** Given P/T nets $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 1, 2$, a P/T net morphism $f : PN_1 \to PN_2$ is given by

$f = (f_P, f_T)$ with functions $f_P : P_1 \to P_2$ and $f_T : T_1 \to T_2$ satisfying

$$\begin{aligned} f_P^{\oplus} \circ pre_1 &= pre_2 \circ f_T \\ f_P^{\oplus} \circ post_1 &= post_2 \circ f_T \end{aligned}$$

Definition 3.4 (Injective, (Jointly) Surjective and Bijective P/T Net Morphisms)**.** A P/T net morphism $f : PN_1 \to PN_2$ with $f = (f_P, f_T)$ is called injective (resp. surjective, bijective) if and only if f_P and f_T are injective (resp. surjective, bijective) functions.
A pair of P/T net morphisms $f_i = (f_{i_P}, f_{i_T}) : PN_i \to PN_3$ with $i = 1, 2$ is called jointly surjective if and only if (f_{1_P}, f_{2_P}) and (f_{1_T}, f_{2_T}) are jointly surjective functions.

Fact 3.5 (Category **PTNet** is a Weak Adhesive HLR Category)**.** P/T nets and P/T net morphisms form the category **PTNet**, where the composition of morphisms is defined componentwise for places and transitions. This category is a weak adhesive HLR category (see Definition 2.3) with the special morphism class $\mathcal{M}$ of all monomorphisms (see Fact 4.21 in [EEPT06]).

Note. To simplify matters, some indices of **PTNet**-morphisms are neglected in the following.

3.1.2 P/T Systems and the Category of P/T Systems

A P/T system is a P/T net with an initial marking and a P/T morphism is a P/T net morphism with an additional property with respect to the marking. This subsection contains a short review of P/T systems with the most important definitions. Finally, the weak adhesive HLR category of P/T systems and P/T morphisms is introduced.

Definition 3.6 (P/T System)**.** A P/T system $PS = (PN, M)$ is a P/T net with an initial marking $M \in P^{\oplus}$.

Definition 3.7 (P/T Morphism (**PTSys**-Morphism))**.** Given P/T systems $PS_i = (PN_i, M_i)$ with $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 1, 2$, a P/T morphism $f : PS_1 \to PS_2$ is given by a P/T net morphism $f = (f_P, f_T)$ with the following additional property:

$$\forall p \in P_1 : M_1(p) \leq M_2(f_P(p))$$

Definition 3.8 (Injective, (Jointly) Surjective, Bijective P/T Morphisms)**.** The definitions of injectivity, (jointly) surjectivity and bijectivity of P/T morphisms are equal to the definitions of the corresponding properties of P/T net morphisms (see Definition 3.4).

Towards the weak adhesive HLR category **PTSys** of P/T systems and P/T morphisms, a special monomorphism class named *strict P/T morphisms*, formalized in the next two definitions, is required.

Definition 3.9 (Marking Strict P/T Morphism)**.** Given P/T systems $PS_i = (PN_i, M_i)$ with $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 1, 2$, a marking strict P/T morphism $f : PS_1 \to PS_2$ is a P/T morphism $f = (f_P, f_T)$ with the following additional property:

$$\forall p \in P_1 : M_1(p) = M_2(f_P(p))$$

Definition 3.10 (Strict P/T Morphism)**.** A *strict* P/T morphism is an injective and *marking strict* (see Definition 3.9) P/T morphism.

Fact 3.11 (Category **PTSys** is a Weak Adhesive HLR Category)**.** P/T systems and P/T morphisms form the category **PTSys**, where the composition of morphisms is defined as composition of P/T net morphisms. This category is a weak adhesive HLR category (see Definition 2.3) with the special morphism class $\mathcal{M}_{strict}$ of all strict (see Definition 3.10) P/T morphisms (proof in Section 4 in [EEH$^+$07]).

Since **PTSys** ia a weak adhesive HLR category with the special morphism class of all *strict morphisms*, it is possible to formulate transformation rules in **PTSys** with two strict morphisms. Referring to the title of this thesis, the next definition introduces a reconfigurable P/T system.

Definition 3.12 (Reconfigurable P/T System)**.** Given a P/T system PS and a set $RULES$ of rules, a reconfigurable P/T system is defined by $(PS, RULES)$.

The following definitions introduce *minimal surjective* and *minimal jointly surjective* P/T morphisms. These morphism classes are required for the proof that **PTSys** with some special morphism classes is a weak adhesive HLR category with NACs (see Definition 2.4). This proof is located in Section 3.2. The morphism classes, formalized in the next definitions, are called *minimal*, although they are defined by the maximal marking of the source places. This is because the marking of the target P/T system is minimal, i.e. every marking that is smaller than the minimal marking violates the well-definedness of the P/T morphism.

Definition 3.13 (Minimal Surjective P/T Morphism)**.** A surjective P/T morphism (see Definition 3.4) $f = (f_P, f_T) : PS_1 \to PS_2$ with $PS_i = (P_i, T_i, pre_i, post_i, M_i)$ for $i = 1, 2$ is called *minimal* if and only if M_2 is minimal, i.e. the following statement holds for all $p_2 \in P_2$:

$$M_2(p_2) = max(\{M_1(p) | p \in f_P^{-1}(p_2)\})$$

Definition 3.14 (Minimal Jointly Surjective P/T Morphisms)**.** A pair of jointly surjective P/T morphisms (see Definition 3.4) $f_i = (f_{i_P}, f_{i_T}) : PN_i \to PN_3$ with $i = 1, 2$ with $PN_j = (P_j, T_j, pre_j, post_j, M_j)$ for $j = 1, 2, 3$ is called *minimal* if and only if M_3 is minimal, i.e. the following statement holds for all $p_3 \in P_3$:

$$M_3(p_3) = max(\{M_1(p) | p \in f_{1_P}^{-1}(p_3)\} \cup \{M_2(p) | p \in f_{2_P}^{-1}(p_3)\})$$

3.2 P/T Systems as weak Adhesive HLR Category with NACs

In this section is shown, that the category **PTSys** is a weak adhesive HLR category with NACs.

Theorem 3.15 ((**PTSys**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) is a Weak Adhesive HLR Category with NACs)**.** *The category* (**PTSys**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) *with the following morphism classes is a weak adhesive HLR category with NACs:*

$\mathcal{M}$	:	*strict P/T morphisms (see Definition 3.10))*
$\mathcal{M}'$	:	*injective P/T morphisms (see Definition 3.8 and Facts A.16 and A.2)*
$\mathcal{Q}$	:	*injective P/T morphisms (see Definition 3.8 and Facts A.16 and A.2)*
$\mathcal{E}'$	:	*minimal jointly surjective P/T morphisms (see Definition 3.14)*

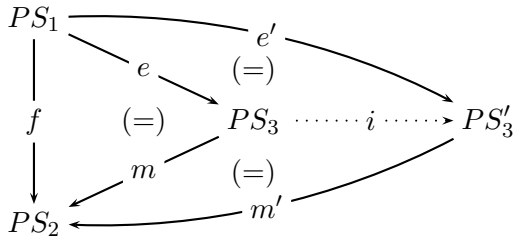
Proof. As already mentioned, (**PTSys**,$\mathcal{M}$) is a weak adhesive HLR category (see Fact 3.11). Therefore, merely the additional properties of a *weak adhesive HLR category with NACs* (see Definition 2.4) need to be proven. These properties are proven in the following Facts 3.17, 3.19, 3.22, 3.23, 3.25, 3.26, 3.27, 3.28, 3.29, 3.30 and 3.31. □

Remark 3.16 (Morphism Class $\mathcal{Q}$). It is an extremely important result that $\mathcal{Q}$-morphisms do not have to be marking strict (see Definition 3.9). This is very useful for modeling rules with NACs. So most restrictions can be expressed with a finite set of NACs that would cause an infinite set of NACs if $\mathcal{Q}$-morphisms were marking strict. This is illustrated in the case study in Section 3.3.

Fact 3.17 ((**PTSys**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Unique Epi-$\mathcal{M}$ Factorization)**.** See Definition 2.5.

Proof. Given P/T systems $PS_i = (PN_i, M_i)$ with $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 1, 2$ and **PTSys**-morphism $f : PS_1 \to PS_2$. A P/T morphism is a tuple of functions and a unique epi-mono factorization exists for every function (see Theorem 3.6.1 in [EMC+01]). So a P/T system $PS_3 = (PN_3, M_3)$ with $PN_3 = (P_3, T_3, pre_3, post_3)$ and morphisms $e = (e_P, e_T) : PS_1 \to PS_3$, $m = (m_P, m_T) : PS_3 \to PS_2$ can be constructed such that

- $P_3 = \{p \in P_2 | \exists p_1 \in P_1 : f_P(p_1) = p\}$
- $T_3 = \{t \in T_2 | \exists t_1 \in T_1 : f_T(t_1) = t\}$
- $pre_3 = pre_{2_{|T_3}}$
- $post_3 = post_{2_{|T_3}}$
- $\forall p \in P_3 : M_3(p) = M_2(p)$



e_P, e_T, m_P and m_T are defined by epi-mono factorizations of f_P and f_T in **Sets**. So $m \circ e = f$ with $\forall p \in P_1 : e_P(p) := f_P(p)$, $\forall t \in T_1 : e_T(t) := f_T(t)$ and m_P and m_T are inclusions. e is an epimorphism because e_P and e_T are surjective and m is a monomorphism because m_P and m_T are injective. m is strict by construction. Next, the well-definedness of e and m is shown.

$$\begin{aligned} \forall t \in T_1 : pre_3(e_T(t)) &= pre_2(f_T(t)) \\ &= f_P^{\oplus}(pre_1(t)) \\ &= e_P^{\oplus}(pre_1(t)) \end{aligned}$$

post analogous.

$$\begin{aligned} \forall p \in P_1 : M_1(p) &\leq M_2(f_P(p)) \\ &= M_3(e_P(p)) \end{aligned}$$

$$\begin{aligned} \forall t \in T_3 : pre_2(m_T(t)) &= pre_2(t) \\ &= pre_3(t) \\ &= m_P^{\oplus}(pre_3(t)) \end{aligned}$$

post analogous.

$$\forall p \in P_3 : M_3(p) = M_2(m_P(p))$$

At last, the universal property is shown. Let $PS_3' = (PN_3', M_3')$ with $PN_3' = (P_3', T_3', pre_3', post_3')$ be a P/T system and $e' : PS_1 \to PS_3'$ and $m' : PS_3' \to PS_2$ be P/T morphisms with $m' \circ e' = f$, where e' is an epimorphism and m' is strict. The unique isomorphism $i = (i_P, i_T) : PS_3 \to PS_3'$ with $i \circ e = e'$ and $m' \circ i = m$ is induced (componentwise in **Sets**) since e_P' and e_T' are surjective and m_P' and m_T' are injective functions. Note that i is marking strict because m and m' are also. $i_P \circ e_P = e_P'$ and $i_T \circ e_T = e_T'$ imply that $i \circ e = e'$ (analogously, $m_P' \circ i_P = m_P$ and $m_T' \circ i_T = m_T$ imply that $m' \circ i = m$). Well-definedness of **PTSys**-morphism

$i = (i_P, i_T)$ is implied by this commutativity:

$$\begin{aligned}\forall t_3 \in T_3 : pre'_3(i_T(t_3)) &= pre'_3(i_T(e_T(t_1))) \text{ with } e_T(t_1) = t_3\\ &= pre'_3(e'_T(t_1))\\ &= e'^{\oplus}_P(pre_1(t_1))\\ &= i^{\oplus}_P(e^{\oplus}_P(pre_1(t_1)))\\ &= i^{\oplus}_P(pre_3(e_T(t_1)))\\ &= i^{\oplus}_P(pre_3(t_3))\end{aligned}$$

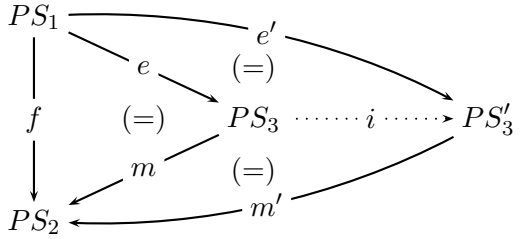
post analogous.
Uniqueness of i follows from the componentwise uniqueness of the functions i_P and i_T. □

According to Remark 5.26 in [EEPT06], $\mathcal{E}'$-$\mathcal{M}'$pair factorization (see Definition 2.6) can be constructed in categories with binary coproducts (see Fact A.17) by $\mathcal{E}_0$-$\mathcal{M}_0$factorization, where $\mathcal{E}_0$ is a class of epimorphisms and $\mathcal{M}_0$ is a class of monomorphisms. The next fact proves that $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has unique $\mathcal{E}_0$-$\mathcal{M}'$ factorization, where $\mathcal{E}_0$ is the class of minimal surjective P/T morphisms (see Definition 3.13) and $\mathcal{M}_0 = \mathcal{M}'$. The later following proof that $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has unique $\mathcal{E}'$-$\mathcal{M}'$ pair factorization uses this fact.

Fact 3.18 ($(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Unique $\mathcal{E}_0$-$\mathcal{M}'$ Factorization)**.** Let $\mathcal{E}_0$ be the class of minimal surjective P/T morphisms (see Definition 3.13).
For every P/T morphism $f : PS_1 \to PS_2$ a unique $\mathcal{E}_0$-$\mathcal{M}'$ factorization (PS_3, e, m) with $m \circ e = f$ exists, where $e \in \mathcal{E}_0$ and $m \in \mathcal{M}'$ with the following universal property:
For all P/T systems PS'_3 and morphisms $e' : PS_1 \to PS'_3$ and $m' : PS'_3 \to PS_2$ with $m' \circ e' = f$, a unique isomorphism $i : PS_3 \to PS'_3$ with $i \circ e = e'$ and $m' \circ i = m$ exists.

Proof. Given P/T systems $PS_i = (PN_i, M_i)$ with $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 1, 2$ and **PTSys**-morphism $f : PS_1 \to PS_2$. Construct P/T system $PS_3 = (PN_3, M_3)$ with $PN_3 = (P_3, T_3, pre_3, post_3)$ and morphisms $e = (e_P, e_T) \in \mathcal{E}_0 : PS_1 \to PS_3$ and $m = (m_P, m_T) : PS_3 \to PS_2$ as described in Fact 3.17 except for M_3:

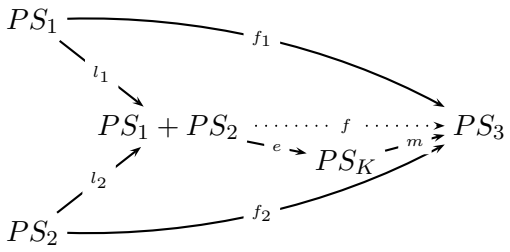
$$\forall p_3 \in P_3 \subseteq P_2 : M_3(p_3) = max(\{M_1(p) | p \in f_P^{-1}(p_3))\}$$



Well-definedness of this construction in **PTNet** is already shown in the proof of Fact 3.17. Marking preserving of e and m as well as $e \in \mathcal{E}_0$ and $m \in \mathcal{M}'$ follow directly by construction.
Finally, the universal property is shown. Let $PS'_3 = (PN'_3, M'_3)$ with $PN'_3 = (P'_3, T'_3, pre'_3, post'_3)$ be a P/T system and $e' \in \mathcal{E}_0 : PS_1 \to PS'_3$ and $m' \in \mathcal{M}' : PS'_3 \to PS_2$ be P/T morphisms with $m' \circ e' = f$. The unique isomorphism $i = (i_P, i_T) : PS_3 \to PS'_3$ with $i \circ e = e'$ and $m' \circ i = m$ is induced in **PTNet** by epi-$\mathcal{M}$ factorization (see Fact 3.17) since $\mathcal{E}_0$-$\mathcal{M}'$ factorization and epi-$\mathcal{M}$ factorization only differ in the construction of M_3 (and M'_3). $M_3 \cong M'_3$ follows directly by the minimal property of e and e'. □

Fact 3.19 ((**PTSys**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Unique $\mathcal{E}'$-$\mathcal{M}'$ Pair Factorization)**.** See Definition 2.6.

Construction. Let $f_1 : PS_1 \to PS_3$ and $f_2 : PS_2 \to PS_3$ with $PS_i = (P_i, T_i, pre_i, post_i, M_i)$ for $i = 1, 2, 3$ be P/T morphisms. Since **PTSys** has binary coproducts (see Fact A.17) there exist inclusions $l_1 : PS_1 \to PS_1 + PS_2$ and $l_2 : PS_2 \to PS_1 + PS_2$ and an induced morphism $f = [f_1, f_2] : PS_1 + PS_2 \to PS_3$ such that $f \circ l_1 = f_1$ and $f \circ l_2 = f_2$, where $PS_1 + PS_2$ is the binary coproduct of PS_1 and PS_2.
Let (PS_K, e, m) (with epimorphism $e \in \mathcal{E}_0$ and $m \in \mathcal{M}'$) be the $\mathcal{E}_0$-$\mathcal{M}$ factorization (see Fact 3.18) of f. $\mathcal{E}'$-$\mathcal{M}'$ pair factorization is defined by $(PS_K, (e_1 = e \circ l_1 : PS_1 \to PS_K, e_2 = e \circ l_2 : PS_2 \to PS_K), m : PS_K \to PS_3)$.



Proof.

1. $m \circ e_1 = m \circ e \circ l_1 = f \circ l_1 = f_1$.

2. $m \circ e_2 = m \circ e \circ l_2 = f \circ l_2 = f_2$.

3. $(e_1, e_2) \in \mathcal{E}'$. (l_1, l_2) are jointly surjective and strict since they are coproduct inclusions (see Fact A.17). The fact that (e_1, e_2) are jointly epimorphic (i.e. jointly surjective) follows directly from the composition property of epimorphisms and jointly epimorphic morphisms (see Fact D.1). $(e_1, e_2) \in \mathcal{E}'$ (are minimal) since $e \in \mathcal{E}_0$ (is minimal) and l_1 and l_2 are strict.

4. $m \in \mathcal{M}'$ follows directly from $\mathcal{E}_0$-$\mathcal{M}'$ factorization.

5. Uniqueness up to isomorphism of $\mathcal{E}'$-$\mathcal{M}'$ pair factorization. Let PS'_K be a P/T system and $e'_1 : PS_1 \to PS'_K$, $e'_2 : PS_2 \to PS'_K$ and $m' \in \mathcal{M}' : PS'_K \to PS_3$, where $(e'_1, e'_2) \in \mathcal{E}'$ and $m' \circ e'_1 = f_1$ and $m' \circ e'_2 = f_2$. Then an induced morphism $[e'_1, e'_2] : PS_1 + PS_2 \to PS'_K$ with $[e'_1, e'_2] \circ l_1 = e'_1$ and $[e'_1, e'_2] \circ l_2 = e'_2$ exists.
   In the following is shown, that $(PS'_K, [e'_1, e'_2], m')$ is the $\mathcal{E}_0$-$\mathcal{M}'$ factorization of f.

$$f \circ l_i = f_i = m' \circ e'_i = m' \circ [e'_1, e'_2] \circ l_i \text{ for } i = 1, 2 \Rightarrow f = m' \circ [e'_1, e'_2]$$

   $[e'_1, e'_2]$ is surjective since (e'_1, e'_2) are jointly surjective by assumption. Strictness of l_1 and l_2 and the fact that $(e'_1, e'_2) \in \mathcal{E}'$ (are minimal) lead to $[e'_1, e'_2] \in \mathcal{E}_0$ (is minimal). $m' \in \mathcal{M}'$ holds by assumption. So $(PS'_K, [e'_1, e'_2], m')$ is the $\mathcal{E}_0$-$\mathcal{M}'$ factorization of f (see Fact 3.18). Uniqueness of this factorization implies the existence of a unique isomorphism $i : PS_K \to PS'_K$ with $i \circ e = [e'_1, e'_2]$ and $m' \circ i = m$. $i \circ e_j = e'_j$ for $j = 1, 2$ remains to be shown.

$$i \circ e_1 = i \circ e \circ l_1 = [e'_1, e'_2] \circ l_1 = e'_1$$
$$i \circ e_2 = i \circ e \circ l_2 = [e'_1, e'_2] \circ l_2 = e'_2$$

□

For showing $\mathcal{M}$-$\mathcal{M}'$ pushout-pullback decomposition property (see Definition 2.7), the use of the fact that **PTNet** with the class $\mathcal{M}$ of all monomorphisms is a weak adhesive HLR category (see Fact 3.5) is convenient. Additionally, a fact expressing the connection between pushouts in category **PTNet** and category **PTSys** is required. This fact is formalized in the next step and used in some of the following proofs.

Fact 3.20 (**PTNet-PTSys** Pushout Equivalence)**.** Given the following commutative diagram (1) in category **PTSys** with $PS_i = (PN_i, M_i)$ with $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 0..3$ in $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ with $m, n \in \mathcal{M}$, then the following statements hold:

1. (1) is a pushout in **PTSys** if (1) is a pushout in **PTNet** and all places in P_1 without a preimage in P_0 are mapped strictly, i.e. $\forall p_1 \in P_1 \backslash m(P_0) : M_1(p_1) = M_3(g(p_1))$.

2. (1) is a pushout in **PTNet** if (1) is a pushout in **PTSys**.

$$\begin{array}{ccc} PS_0 & \overset{m \in \mathcal{M}}{\rightarrowtail} & PS_1 \\ \downarrow f & (1) & \downarrow g \\ PS_2 & \underset{n \in \mathcal{M}}{\rightarrowtail} & PS_3 \end{array}$$

Proof. **Part 1**. Let (1) be a pushout in **PTNet** with $\forall p_3 \in g(P_1) \backslash m(P_0) : M_1(p_1) = M_3(p_3)$, where $p_3 = g(p_1)$. Obviously (1) commutes in **PTSys**. Let $PS_4 = (PN_4, M_4)$ with $PN_4 = (P_4, T_4, pre_4, post_4)$ be a P/T system and $k : PS_1 \to PS_4, h : PS_2 \to PS_4$ be morphisms with $k \circ m = h \circ f$ in **PTSys**. Since (1) is a pushout in **PTNet**, a unique induced morphism $x : PS_3 \to PS_4$ with $x \circ g = k$ and $x \circ n = h$ in **PTNet** exists. Now, $x \in$ **PTSys** is shown, i.e. $M_3(p_3) \leq M_4(x(p_3))$ for all $p_3 \in P_3$.

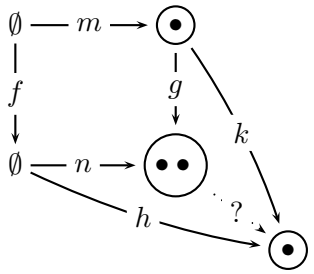
1. For $p_3 = g(p_1)$ with $p_1 \in P_1 \backslash m(P_0)$, it holds that $M_3(p_3) = M_3(g(p_1)) = M_1(p_1) \leq M_4(k(p_1)) = M_4(x(g(p_1))) = M_4(x(p_3))$.
2. For $p_3 = n(p_2)$ with $p_2 \in P_2$, it holds that $M_3(p_3) = M_3(n(p_2)) = M_2(p_2) \leq M_4(h(p_2)) = M_4(x(n(p_2))) = M_4(x(p_3))$.

Uniqueness of x follows from the universal property of pushouts (in **PTNet**).

Part 2. Let (1) be a pushout in **PTSys**. From the construction of pushouts in **PTNet** (see Fact A.12) follows that (1) is a pushout in **PTSys**. □

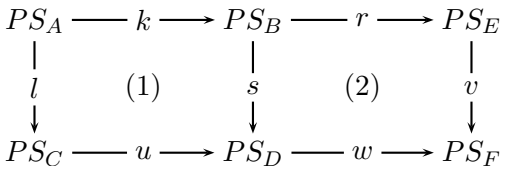
For pointing out this fact, the following counterexample demonstrates that not every pushout in **PTNet** containing two strict morphisms is a pushout in **PTSys**.

Counterexample 3.21 (Not every Pushout in **PTNet** with $m, n \in \mathcal{M}$ is a Pushout in **PTSys**)**.**



Fact 3.22 ((**PTSys**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has $\mathcal{M}$-$\mathcal{M}'$ Pushout-Pullback Decomposition Property)**.** See Definition 2.7.

Proof. Given the following commutative diagram in category **PTSys** with P/T systems $PS_x = (PN_x, M_x)$ with $PN_x = (P_x, T_x, pre_x, post_x)$ for $x = A, B, C, D, E, F$ and $l \in \mathcal{M}$ and $w \in \mathcal{M}'$, where (1+2) is a pushout and (2) a pullback.



$l \in \mathcal{M}$, (1+2) is a pushout, so $v \in \mathcal{M}$ and, additionally, $s \in \mathcal{M}$ since (2) is a pullback and $\mathcal{M}$-morphisms are closed under pushouts and pullbacks (see Def. 2.3). Theorem 4.26 (1.) in [EEPT06] (Pushouts along $\mathcal{M}$-Morphisms are Pullbacks) implies that

(1+2) is a pullback. (1) is also a pullback because of pullback decomposition.
The fact that (1) and (2) are pushouts in **PTSys** remains to be shown. (**PTNet**,$\mathcal{M}'$) is a weak adhesive HLR category, $\mathcal{M} \subset \mathcal{M}'$ and $\mathcal{M}$ Pushout-Pullback Decomposition Lemma (see Theorem 4.26 (2.) in [EEPT06]) is valid in (weak) adhesive HLR categories. So (1) and (2) are pushouts in **PTNet**.
Now, the assumption of Fact 3.20 (1.) is shown, i.e. u is strict on the places of PS_C that have no preimage in P/T system PS_A.
Let $p_C \in P_C \backslash l(P_A)$.

1. $u_{|P_C \backslash (P_A)}$ is injective. Suppose that $\exists p_C \neq p'_C \in P_C \backslash l(P_A) : u(p_C) = u(p'_C)$, then an image $p_F \in P_F$ of p_C and p'_C with $w(u(p_C)) = p_F = w(u(p'_C))$ exists. Pushout (1+2) and $l \in \mathcal{M}$ imply that $\exists p_A, p'_A \in P_A : r(k(v(p_A))) = p_F = r(k(v(p'_A)))$ and, additionally, $l(p_A) = p_C \wedge l(p'_A) = p'_C$, which is a contradiction to the assumption that $p_C \neq p'_C \in P_C \backslash l(p_A)$.

2. $u_{|P_C \backslash (P_A)}$ is mapping strict.
   Since (1+2) is a pushout in **PTSys**, $M_C(p_C) = M_F(w(u(p_C)))$ holds. Additionally, it is valid that

   (a) $M_C(p_C) \leq M_D(u(p_C))$ and
   (b) $M_D(u(p_C)) \leq M_F(w(u(p_C)))$.

$$M_C(p_C) \leq M_D(u(p_C)) \leq M_F(w(u(p_C))) = M_C(p_C)$$
$$\Rightarrow M_C(p_C) = M_D(u(p_C))$$

Fact 3.20 (1.) leads to pushout (1) in **PTSys**. From pushout decomposition follows that (2) is a pushout in **PTSys**. □

Fact 3.23 ((**PTSys**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has $\mathcal{M}$-$\mathcal{Q}$ Pushout-Pullback Decomposition Property)**.** See Definition 2.8. Analogous to Fact 3.22 since $\mathcal{Q}$=$\mathcal{M}'$.

Subsequently, the fact that **PTNet** has initial pushouts over monomorphisms (i.e. $\mathcal{M}'$-morphisms) is shown. Then the fact that (**PTSys**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has initial pushouts over $\mathcal{M}'$-morphisms is proven by applying Fact 3.20 (1.).

Fact 3.24 (**PTNet** has Initial Pushouts over Monomorphisms)**.** See Definition 2.10.

Construction. Let $\mathcal{M}_{inj}$ be the class of all injective P/T net morphisms.
Given an injective **PTNet**-morphism $f : PN_0 \to PN_1$ with $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 0..1$, then the boundary $PN_B = (P_B, \emptyset, pre_B, post_B)$ can be constructed as follows:

- $P_B = \{p \in P_0 | \exists t \in (T_1 \backslash f(T_0)) : f(p) \in \bullet t \cup t\bullet\}$
- $pre_B, post_B : \emptyset \to P_B^{\oplus}$

The context object $PN_C = (P_C, T_C, pre_C, post_C)$ can be constructed as pushout complement (see Section 7.3 in [EP04]):

- $P_C = (P_1 \backslash f(P_0)) \cup f(b(P_B))$
- $T_C = (T_1 \backslash f(T_0)) \cup f(b(T_B)) = (T_1 \backslash f(T_0)) \cup f(b(\emptyset)) = (T_1 \backslash f(T_0))$
- $pre_C = pre_{1_{|T_C}}$
- $post_C = post_{1_{|T_C}}$

The monomorphisms $b : PN_B \to PN_0$ and $c : PN_C \to PN_1$ are inclusions and the monomorphism $f' : PN_B \to PN_C$ can be constructed as follows: $f' = (f'_P, f'_T)$ with $f'_P = f_{P_{|P_B}}$ and $f'_T = f_{T_{|T_B}} = \emptyset$. This construction leads to *initial pushout* (1).

$$\begin{array}{ccc} PN_B & \xrightarrow{b} & PN_0 \\ \downarrow{\scriptstyle f'} & (1) & \downarrow{\scriptstyle f} \\ PN_C & \xrightarrow{c} & PN_1 \end{array} \qquad \begin{array}{ccccccc} & & & b & & & \\ & \curvearrowright & & & & \curvearrowright & \\ PN_B & \xrightarrow{b^*} & PN_2 & & \xrightarrow{b'} & & PN_0 \\ \downarrow{\scriptstyle f'} & (3) & \downarrow{\scriptstyle f''} & & (2) & & \downarrow{\scriptstyle f} \\ PN_C & \xrightarrow{c^*} & PN_3 & & \xrightarrow{c'} & & PN_1 \\ & \searrow & & c & & \searrow & \end{array}$$

Proof. First of all, the well-definedness of the constructions above and the fact that (1) is a pushout are shown. PN_B is a valid P/T net since T_B is empty and pre_B and $post_B$ are the empty function (which is unique). The gluing condition (see Section 7.3 in [EP04]) for $PN_B \xrightarrow{b} PN_0 \xrightarrow{f} PN_1$ is always satisfied by construction since IP=$\emptyset$, $DP \subseteq GP$. Hence, the context net PN_C and the morphisms f' and c exist and are well-defined. Additionally, (1) is a pushout. Since f is a monomorphism, f' is also a monomorphism by construction. b, c are monomorphisms since they are inclusions.

Next, the initiality of (1) is proven. In the first step, the existence of b^* and c^* is shown. Given pushout (2) with monomorphism b'. Then c' is also a monomorphism ($\mathcal{M}$ is closed under pushouts and $(\mathbf{PTNet}, \mathcal{M}_{inj})$ is a weak adhesive HLR category) and f'' is a monomorphism by construction. The fact that the gluing condition is satisfied if and only if the pushout complement exists (see Section 3.5 and Section 7.3 in [EP04]) implies that the gluing condition for $PN_2 \xrightarrow{b'} PN_0 \xrightarrow{f} PN_1$ is satisfied and $P_B = DP \subseteq P_2$ holds, i.e. $b(PN_B) \subseteq b'(PN_2)$. Analogously, because of the uniqueness and the construction of pushout complements, $c(PN_C) \subseteq c'(PN_3)$ holds. Obviously, induced morphisms $b^* : PN_B \to PN_2$ with $\forall p_B \in P_B : b^*(p_B) = p_2$, where $b'(p_2) = b(p_B)$, and $c^* : PN_C \to PN_3$ with $\forall n_C \in P_C \uplus T_C : c^*(n_C) = n_3$, where $c'(n_3) = c(n_C)$, exist. So $b' \circ b^* = b$ and $c' \circ c^* = c$. Since b', b, c' and c are monomorphisms, b^* and c^* are also monomorphisms.

Now, the well-definedness of b^* and c^* is shown. The proof for morphism b^* is

trivial since $T_B = \emptyset$.

$$\begin{array}{rrcll} & c'(pre_3(c^*(t_C))) & = & c'(pre_3(t_3)) & \text{with } c'(t_3) = c(t_C)\ (*) \\ & & = & pre_1(c'(t_3)) & c' \text{ is well-defined} \\ & & = & pre_1(c(t_C)) & \text{see } (*) \\ & & = & c(pre_C(t_C)) & c \text{ is well-defined} \\ & & = & c'(c^*(pre_C(t_C))) & c' \circ c^* = c \\ \Rightarrow & pre_3(c^*(t_C)) & = & c^*(pre_C(t_C)) & c' \text{ is mono} \end{array}$$

The last statement holds because of the well-definedness of c'. The proof for *post* is analogous.

Finally, the fact that (3) is a pushout remains to be proven. In the first step, only the commutativity of (3) is shown. Recapitulating, the following statements hold: $b' \circ b^* = b$, $c' \circ c^* = c$, $f \circ b' = c' \circ f''$, $f \circ b = c \circ f'$.

$$\begin{array}{rrcll} & c' \circ f'' \circ b^* & = & f \circ b' \circ b^* & \text{PO (2)} \\ & & = & f \circ b & b' \circ b^* = b \\ & & = & c \circ f' & \text{PO (1)} \\ & & = & c' \circ c^* \circ f' & c' \circ c^* = c \\ \Rightarrow & f'' \circ b^* & = & c^* \circ f' & c' \text{ is mono} \end{array}$$

Now, the fact that the commutative diagram (3) is a pushout, i.e. the universal property holds, is shown. Remember that pushouts in **PTNet** are constructed componentwise (see Fact A.12). In consequence of Fact D.3, the following remains to be shown:

1. Every place and every transition n_3 in PN_3 that has preimages in PN_2 as well as in PN_C has also a preimage n_B in PN_B with $f''(b^*(n_B)) = n_3 = c^*(f'(n_B))$.

2. (f'', c^*) are jointly surjective.

To simplify matters, the following proof is accomplished for places and transitions together. Let $n_3 \in N_3$ for N being P or T.

1. $n_3 \in f''(N_2) \cap c^*(N_C)$. Let $n_2 \in N_2$ with $f''(n_2) = n_3$ and $n_C \in N_C$ with $c^*(n_C) = n_3$ and assume $\nexists n_B \in N_B : b^*(n_B) = n_2 \wedge f'(n_B) = n_C$. Then $b'(n_2) \in N_0$ and, since (2) is a pushout, $c'(n_3) = f(b'(n_2))$, i.e. $c'(n_3) = c'(c^*(n_C)) = c(n_C)$. $c(n_C) = f(b'(n_2))$ and pushout (3) imply that $\exists n_B \in N_B : (b(n_B) = b'(n_2) \wedge f'(n_B) = n_C)$, which is a contradiction to the assumption that $\nexists n_B \in N_B : b^*(n_B) = n_2 \wedge f'(n_B) = n_C$.

2. $n_3 \notin f''(N_2) \cup c^*(N_C)$. Then $c'(n_3) \in N_1$ and, since (2) is a pushout, there are two possibilities:

   (a) $\exists n_0 \in N_0 : f(n_0) = c'(n_3)$. Pushout (2) implies that $\exists n_2 \in N_2 : (b'(n_2) = n_0 \wedge f''(n_2) = n_3)$, which is a contradiction to the assumption that $n_3 \notin f''(N_2)$.

(b) $\nexists n_0 \in N_0 : f(n_0) = c'(n_3)$. Pushout (1) implies that $\exists n_C \in N_C : c(n_C) = c'(n_3)$ and $c' \circ c^* = c$ leads to $c^*(n_C) = n_3$, which is a contradiction to the assumption that $n_3 \notin c^*(N_C)$.

By Fact D.3, (3) is a componentwise pushout in **Sets**, i.e. (3) is a pushout in **PTNet**. □

Fact 3.25 ((**PTSys**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Initial Pushouts over $\mathcal{M}'$-Morphisms)**.** See Definition 2.10.

Construction. Given an injective **PTSys**-morphism $f : PS_0 \to PS_1 \in \mathcal{M}'$ with $PS_i = (PN_i, M_i)$ and $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 0..1$, then the boundary $PS_B = (PN_B, M_B)$ with $PN_B = (P_B, \emptyset, pre_B, post_B)$ can be constructed analogously to the boundary net in **PTNet** except for the set P_B.

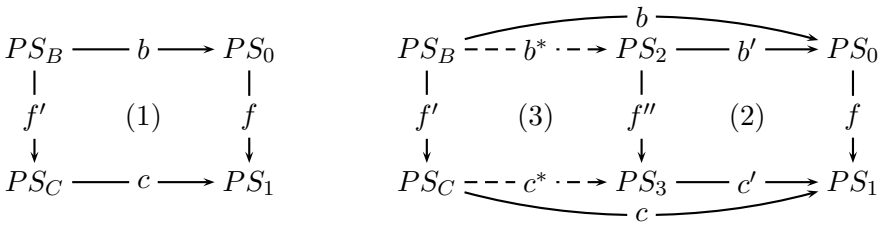
- $P_B = \{p \in P_0 | \exists t \in (T_1 \backslash f(T_0)) : f(p) \in {\bullet t} \cup t^{\bullet}\} \cup \{p \in P_0 | M_0(p) < M_1(f(p))\}$
- $\forall p_B \in P_B \subseteq P_0 : M_B(p_B) = M_0(p_B)$

Note that the boundary net in **PTSys**, compared to to the boundary net in **PTNet**, contains additionally all places that are not mapped strictly by f. These places form the set $NSP = \{p \in P_0 | M_0(p) < M_1(f(p))\}$ of the *non strict places*.

The context object $PS_C = (PN_C, M_C)$ with $PN_C = (P_C, T_C, pre_C, post_C)$ can be constructed as pushout complement in **PTSys** (see Theorem 5 in [EEH+07]), analogous to the context net in **PTNet**.

- $\forall p_C \in P_C \subseteq P_1 : M_C(p_C) = M_1(p_C)$

The construction of the strict morphisms $b : PS_B \to PS_0 \in \mathcal{M}$, $c : PS_C \to PS_1 \in \mathcal{M}$ and the morphism $f' : PS_B \to PS_C \in \mathcal{M}'$ is analogous to the construction in **PTNet**. This construction leads to *initial pushout* (1).



Proof. The gluing condition for **PTSys** (see Def. A.21) contains only one additional requirement compared to the gluing condition for **PTNet**: Places to be deleted have to be mapped strictly. Since P_B contains all places of the set NSP, i.e. the places that are not mapped strictly, this additional condition is always satisfied. The set P_B is the minimal set of places, that satisfies this condition. So it can be embedded in every interface PS_2 of pushout (2) (see Theorem 5 in [EEH+07]). The same statement holds for PS_C since it is constructed as pushout complement. The proof

is analogous to the proof for **PTNet**.
The fact that (3) is a pushout in **PTSys** remains to be shown. According to Fact 3.20 (1.), it is sufficient to show that all places in P_2 which have no preimage in P_B are mapped strictly by f''. Note that b^* and c^* are strict since b and c are strict by construction and b' and c' are strict because of assumption.
Let $p_2 \in P_2 \backslash b^*(P_B)$. Suppose that $M_2(p_2) < M_3(f''(p_2))$. Then $b'(p_2) \in P_0$ with $M_2(p_2) = M_0(b'(p_2))$ since $b' \in \mathcal{M}$. Pushout (2) implies that $\exists p_1 \in P_1 : c'(f''(p_2)) = p_1 = f(b'(p_2))$ and, because of the strictness of c', $M_3(f''(p_2)) = M_1(p_1)$, i.e. $M_0(b'(p_2)) < M_1(f(b'(p_2)))$. The construction of PS_B implies that $\exists p_B \in P_B : b(p_B) = b'(p_2)$. $b' \circ b^* = b$ leads to $b^*(p_B) = p_2$, which is a contradiction to the assumption that $p_2 \in P_2 \backslash b^*(P_B)$. That means every place in P_2 that has no preimage in P_B is mapped strictly by f''. By Fact 3.20 (1.), (3) is a pushout in **PTSys**. □

Fact 3.26 ($\mathcal{M}'$ in $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is closed under POs and PBs along $\mathcal{M}$-Morphisms)**.** See Definition 2.11.

Proof. **Pushout.** This follows from closure property of $\mathcal{M}$-morphisms along pushouts and pullbacks in (weak) adhesive HLR categories since (**PTNet**,$\mathcal{M}'$) is a weak adhesive HLR category and Fact 3.20 (2.) holds.
Pullback. This follows from standard category theory (pullbacks preserve monomorphisms). □

Fact 3.27 ($\mathcal{Q}$ in $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is closed under POs and PBs along $\mathcal{M}$-Morphisms)**.** See Definition 2.12. Analogous to Fact 3.26 since $\mathcal{Q}$=$\mathcal{M}'$.

Fact 3.28 ($(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Induced Pullback-Pushout Property for $\mathcal{M}$ and $\mathcal{Q}$)**.** See Definition 2.13.

Proof. Given the following pullback and pushout with $PS_i = (PN_i, M_i)$ and $PN_i = (P_i, T_i, pre_i, post_i)$ for $i = 0..4$, $h \in \mathcal{M}$ and $h' \in \mathcal{Q}$ in $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$. In the following is shown, that the induced morphism $x : PS_3 \to PS_4$ with $x \circ g' = h$ and $x \circ f' = h'$ is a monomorphism in $\mathcal{Q}$.

$$\begin{array}{ccc} PS_0 & \xrightarrow{f} & PS_1 \\ {\scriptstyle g}\downarrow & (PB) & \downarrow{\scriptstyle h} \\ PS_2 & \xrightarrow{h'} & PS_4 \end{array} \qquad \begin{array}{ccc} PS_0 & \xrightarrow{f} & PS_1 \\ {\scriptstyle g}\downarrow & (PO) & \downarrow{\scriptstyle g'} \\ PS_2 & \xrightarrow{f'} & PS_3 \end{array}$$

Pushouts and pullbacks along $\mathcal{M}$-morphisms can be constructed componentwise (see Section 4 in [EEH+07]) and $h \in \mathcal{M}$ and $h' \in \mathcal{Q}$ (are monomorphisms). Since $\mathcal{M}$-morphisms are closed under pushouts and pullbacks, $g \in \mathcal{M}$ and $g' \in \mathcal{M}$ follow. Since pullbacks preserve monomorphisms and $h' \in \mathcal{Q}$ (is a monomorphism), $f \in \mathcal{Q}$ (is also a monomorphism). Fact 3.27 implies that $f' \in \mathcal{Q}$ (is also a monomorphism). For the following part, it is important to consider that (g', f') are jointly surjective due to pushout (PO). To simplify matters, the following proof is accomplished for

places and transitions together. Let N be P or T.
Suppose that $x : PS_3 \to PS_4 \notin \mathcal{Q}$, i.e. x is not injective, i.e. $\exists n_3 \neq n_3' \in N_3 : x(n_3) = x(n_3')$. Let $n_4 = x(n_3) = x(n_3') \in N_4$, some cases can be distinguished:

1. $n_3, n_3' \in g'(N_1)$. g' being injective and $x \circ g' = h$, yield directly that h is not injective. This contradicts to the fact that h is a monomorphism.

2. $n_3 \in g'(N_1)$ and $n_3' \in f'(N_2)$. Let $n_1 \in N_1$ and $n_2 \in N_2$ with $g'(n_1) = n_3$ and $f'(n_2) = n_3'$. $x \circ g' = h$ and $x \circ f' = h'$ lead to the fact that $h(n_1) = n_4 = h'(n_2)$. Pullback (PB) implies the existence of $n_0 \in N_0$ with $f(n_0) = n_1$ and $g(n_0) = n_2$, contradicting to the fact that (PO) is a pushout.

Since the construction is symmetrical and (g', f') are jointly surjective, all additional cases can be reduced to these two. □

Fact 3.29 (($\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Composition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$)**.** See Definition 2.14.

Proof. Let $f : A \to B \in \mathcal{Q}$ and $g : B \to C \in \mathcal{M}'$ be **PTSys**-morphisms. From standard category theory follows that $g \circ f \in \mathcal{Q}$ (since the composition of monomorphisms is a monomorphism). □

Fact 3.30 (($\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Decomposition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$)**.** See Definition 2.15.

Proof. Let $g \circ f \in \mathcal{Q}$ and $g \in \mathcal{M}'$ be **PTSys**-morphisms. From standard category theory follows that $f \in \mathcal{Q}$ (decomposition property of monomorphisms). □

Fact 3.31 ($\mathcal{Q}$ in ($\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) is closed under Composition and Decomposition)**.** See Definition 2.16. Analogous to Fact 3.29 and Fact 3.30 since $\mathcal{Q}$=$\mathcal{M}'$.

3.3 Case Study: Airport Control System

3.3.1 Overview

In this chapter, an example for a reconfigurable P/T system with negative application conditions is presented. The example is an airport control system, called *ACS*, especially designed for small but expandable airports. This system should prevent accidents in the airport area. ACS is not designed as an automated control system but rather as an assistance for the tower employees. It has to secure that some areas of the airport, like the starting or the landing runway, can only be used exclusively by a given amount of airplanes. However, the system is not responsible for the coordination of the airplanes at the gates. While running, ACS can adapt to various changes of the airport. This is extremely helpful if an airport rearranges because the whole system does not need to be stopped and upgraded.

ACS is precisely described and modeled as reconfigurable P/T system with negative application conditions in this chapter. Every firing of a transition represents a process at the airport and every transformation of the P/T system reflects the rearrangement of the airport. In the context of this example, a lot of important general results of adhesive HLR systems with NACs can be applied.

The example is divided into different parts. First of all, a detailed description of the system is given. It contains a requirement engineering of ACS. The underlying P/T systems and rules of the transformation system are presented afterwards.

3.3.2 Detailed Description

In the basic ACS only one runway is available which is used both as starting and as landing runway. An airplane can only receive landing or starting permit if the runway is not in use by another airplane. For ACS, the airport consists of the mentioned runway, the boarding and the deboarding area, the arrival and the departure airspace of the airport, as well as the tower. The tower is only responsible for managing the exclusive use of the different airport areas. In the context of ACS, airports with a shared runway for starting and landing airplanes are called level-1-airports. Airports that have separate starting and landing runways are called level-2-airports. ACS provides a variety of different expansions for airports. Possible expansions are increasing and decreasing the number of gates, adding and removing a starting or landing runway, adding a hangar and adding a maintenance hangar. A more detailed description of these expansions follows directly.

The increase of the number of gates offers more space for airplanes at the gates, so that several airplanes can accomplish the boarding and deboarding procedure simultaneously. This provides steady traffic at the airport. By applying this expansion, the maximum number of airplanes at the boarding and deboarding area raises by one. ACS has no limitation for the number of gates. This expansion can be revoked by decreasing the number of gates. Even though, the last gate of the airport cannot be removed.
Additional starting runways are also useful expansions because they support the steady traffic at the airport. For safety reasons, adding a starting runway to a level-1-airport automatically changes the shared runway of the airport into a landing runway. The application of this expansion to a level-2-airport does not affect the existing landing runways. The number of runways is not limited by ACS. Airplanes are able to takeoff simultaneously at different starting runways. If there are enough positions at the gates, the airplanes may also enter the deboarding area concurrently. This expansion can be revoked by removing a starting runway. Therefore, several conditions have to be fulfilled:

- The starting runway is not in current use.
- The airport is a level-2-airport, i.e. a shared runway cannot be removed.

- If this starting runway is the last starting runway of the airport, it can only be removed if exactly one landing runway exists. In this case, the last landing runway automatically changes into a shared runway.

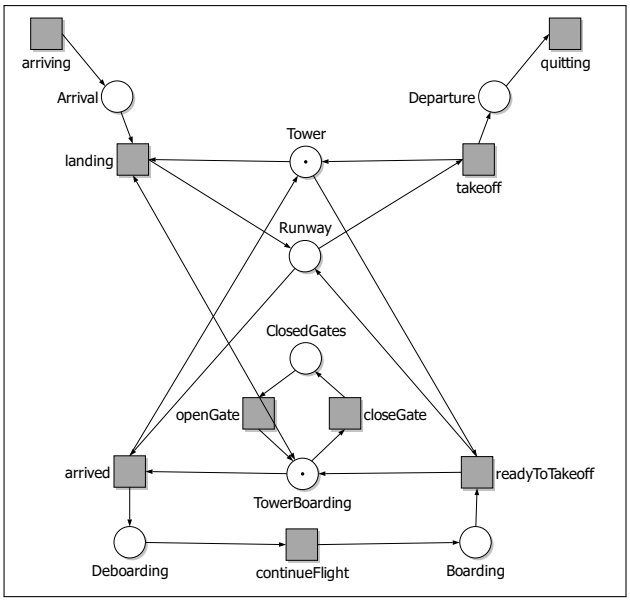
This means, before removing the last starting runway of an airport with multiple landing runways, a landing runway has to be transformed into a starting runway or an additional starting runway has to be added. In this example, the mentioned transformation of a landing runway into a starting runway is modeled by removing a landing runway and adding a new starting runway.
The removing of the last starting runway represents the case of an emergency, for example when the last starting runway is dilapidated. It is only possible to remove the last starting runway under the assumption that only one single landing runway exists because in the case of the existence of multiple landing runways one of them could be turned into a starting runway.
Additional landing runways offer exactly the same advantages and the same expanding possibilities as additional starting runways.

Another possible expansion of the airport is the construction of a hangar which provides space for up to five airplanes. The hangar can be reached directly from the deboarding area and, if existent, from the maintenance hangar. Airplanes are only allowed to enter the hangar if there are enough positions available. Leaving the hangar, the airplanes can reach the boarding area directly and also, if existing, the maintenance hangar. Advantages of a hangar are the increased capacity of airplanes at the airport and the possibility to store airplanes temporarily, so they do not block the area of the gates and the air traffic can be continued more easily.
Another alternative expansion is the construction of a maintenance hangar in which airplanes are repaired and maintained. In contrast to the hangar, the maintenance hangar can only be used exclusively by one airplane. The maintenance hangar can be reached directly from the deboarding area and, if existing, from the hangar. Airplanes are only able to enter the maintenance hangar if it is not in current use. Leaving the maintenance hangar, the airplanes can reach the boading area directly and also, if existing, the hangar.
To simplify matters, the hangar and the maintenance hangar are not expandable or removable.

3.3.3 Systems and Rules

$ACS = (Lvl1ap, ACS\text{-}RULES)$ is a reconfigurable P/T system with NACs with the following P/T system $Lvl1ap$ and the following set of rules with NACs $ACS\text{-}RULES$. For simplicity, hooked arrows always represent inclusions. So the morphisms do not need to be defined explicitly every time.

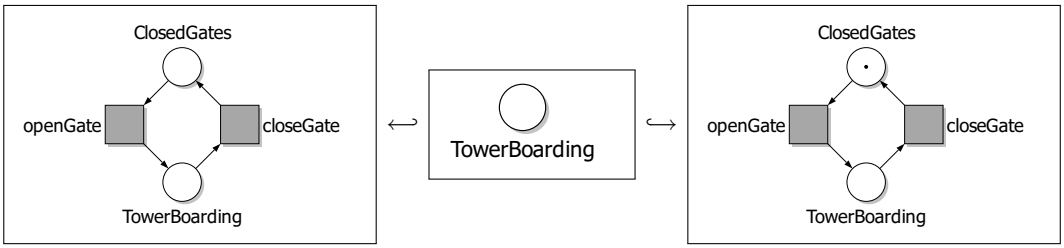
Startsystem: *Lvl1ap*



This P/T system represents a level-1-airport. Since airplanes are able to arrive at the airport anytime, the transition *arriving* has an empty pre-domain and is always enabled. The place *Arrival* contains a token for every arriving airplane that is in the airspace of the airport. Exclusively use of the runway by only one airplane is guaranteed by the place $Tower$, where the initial marking is one token. Landing permit for an airplane of the place *Arrival* can only be granted if the transition *landing* is enabled. This means, the runway is not in current use and enough positions are available at the gate area. The gate area consists of the boarding and the deboarding area, i.e. the places *boarding* and *deboarding*. It is an important condition for arriving airplanes that enough positions are available at the gate area. This is guaranteed by the edges between transition *landing* and place $TowerBoarding$. If this condition is not fulfilled, an arriving airplane could not leave the shared runway and pass through the gate area. It would block the shared runway and no other airplane at the gate area could takeoff. However, this situation would not be a deadlock because the arrived airplane could takeoff and leave the airport immediately. Nevertheless, it seems useless that an airplane could get landing permit if there are not enough positions available. This condition for granting landing permit disappears for level-2-airports since they have separate starting and landing runways. This way, an arriving airplane cannot block a starting airplane. Therefore, the mentioned edges between transition *landing* and place $TowerBoarding$ are not required in level-2-airports.
An airplane token passes through the places *Runway*, *Deboarding*, *Boarding* and *Runway* again until it leaves the airport by firing transition $takeoff$. Afterwards, another airplane can receive landing permit. Every token at the place *Departure* represents a started airplane in the airspace of the airport. Leaving the airspace of

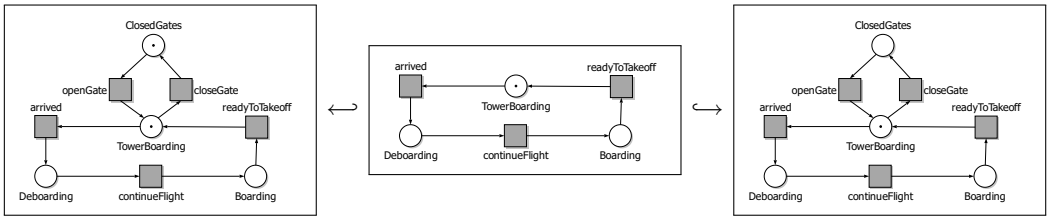
the airport is realized by firing transition *quitting*.
The place *ClosedGates* and the transitions *openGate* and *closeGate* offer the option to open and close gates. This place is used for changing the number of gates, independent of the number of runways. The transitions *openGate* and *closeGate* should only be fired before the number of gates is changed by a rule. They are not required for the regular operation of the airport.

Rule 1: $increaseNumberOfGates$



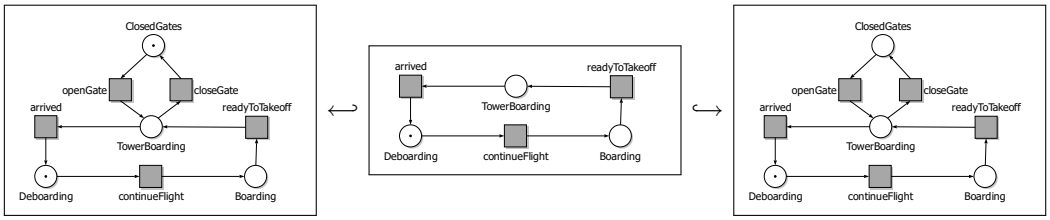
Increasing the number of gates by one is represented by this rule. It removes place *ClosedGates* and the transitions *openGate* and *closeGate* and replaces them with transitions having the name as before and a new place *ClosedGates*, containing a token. This complicated procedure for adding a token is necessary since the morphisms l and r of a rule $L \xleftarrow{l} K \xrightarrow{r} R$ have to be marking strict. Therefore, it is not possible to change the amount of tokens of an existing place. Although, it is possible, as modeled in this rule, to remove a place without tokens and to add a new place (with the same name) with tokens. Note that a consistent match is marking strict on places to be deleted (see gluing condition for P/T systems in Fact A.21), i.e. it is guaranteed that there are no tokens at place *ClosedGates* of the (source) airport net.

Rule 2a: $decreaseNumberOfGates1$

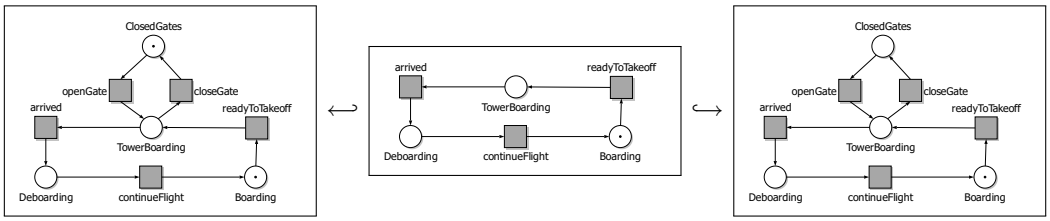


To decrease the number of gates the use of the inverse production of rule 1 is not sufficient because of the additional condition that the last gate cannot be removed. Therefore, at least one token has to be at one of the places $TowerBoarding$, *Boarding* or *Deboarding*. To ensure this condition three rules are required (2a, 2b, 2c). The functionality of these rules are equal and correlate with the inverse production of rule 1 with the mentioned additional condition.

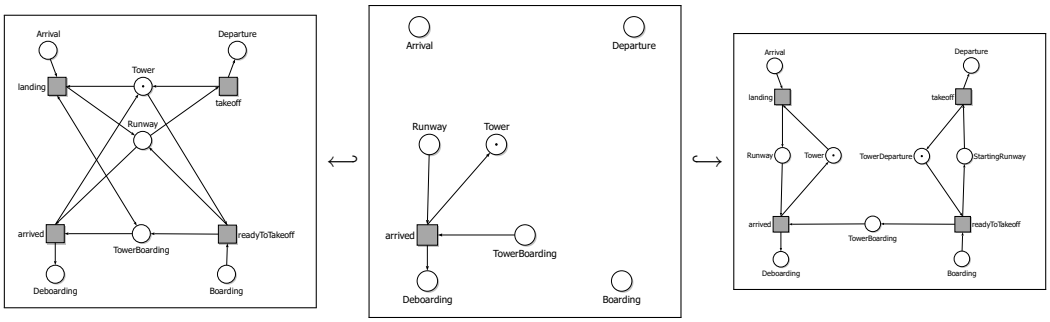
Rule 2b: *decreaseNumberOfGates2*



Rule 2c: *decreaseNumberOfGates3*

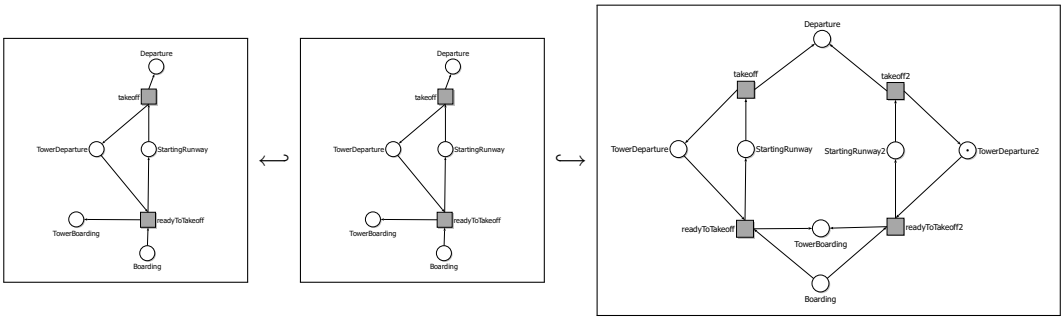


Rule 3a: *addSeperateStartingRunway*



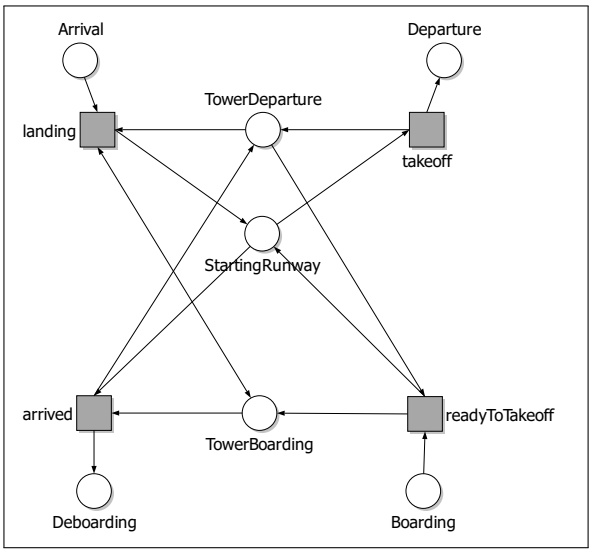
This rule can only be applied to a level-1-airport. A seperate starting runway is added and the shared runway of the airport is automatically changed into a landing runway. The token at the place *Tower* prevents this rule from being applied to an airport where an airplane is on the runway. This is necessary because ACS has no information about the airplanes, i.e. it is not known if an airplane on the shared runway is arriving or departing.

Rule 3b: *addStartingRunway*



Adding an additional starting runway is modeled by this rule. The left-hand side of this rule contains an existing starting runway to prevent nonsensical matches. Alternatively, adding labels to places would prevent a lot of nonsensical matches and the nets of the rules would be smaller. Labels are introduced in Chapter 5 of this thesis and their advantages are presented later. So they are not used in this example. To ensure that this rule can only be applied to a level-2-airport, a negative application condition is required.

Rule 3b - NAC 1:



Through this negative application condition it is not possible to apply this rule to a level-1-airport.

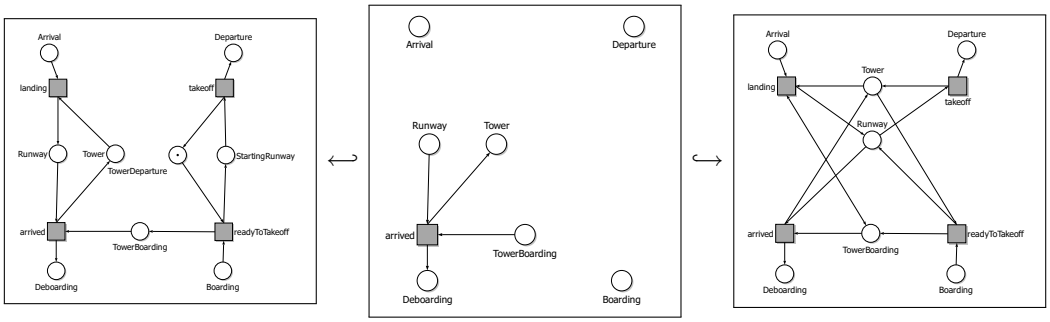
Remark 3.32 (Morphism Class $\mathcal{Q}$). As already mentioned in Section 3.16 the fact that $\mathcal{Q}$-morphisms do not have to be marking strict is an important result for reconfigurable P/T systems. So one NAC is sufficient for this rule. Marking strictness of $\mathcal{Q}$-morphisms would cause an infinite set of NACs for this rule: For every possible marking of the places one NAC would be necessary.
This also holds for all remaining rules.

Rule 4a: *removeStartingRunway*

Inverse to rule 3b without NACs.

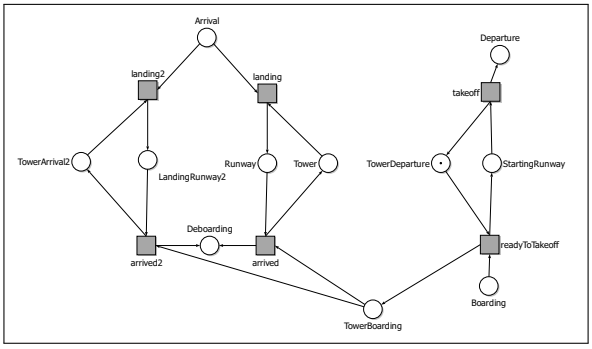
This rule removes an empty starting runway of a level-2-airport with multiple starting runways. It does not require a negative application condition because its left-hand side contains two starting runways.

Rule 4b: *removeLastStartingRunway*



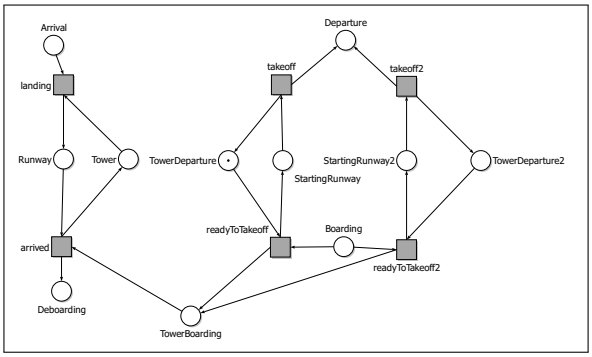
Rule *removeLastStartingRunway* removes the last starting runway of a level-2-airport under the assumption that this starting runway is empty and only one landing runway exist. Additionally, the last existing landing runway is automatically changed into a shared runway. Negative application conditions are required to prevent the existence of additional runways.

Rule 4b - NAC 1:



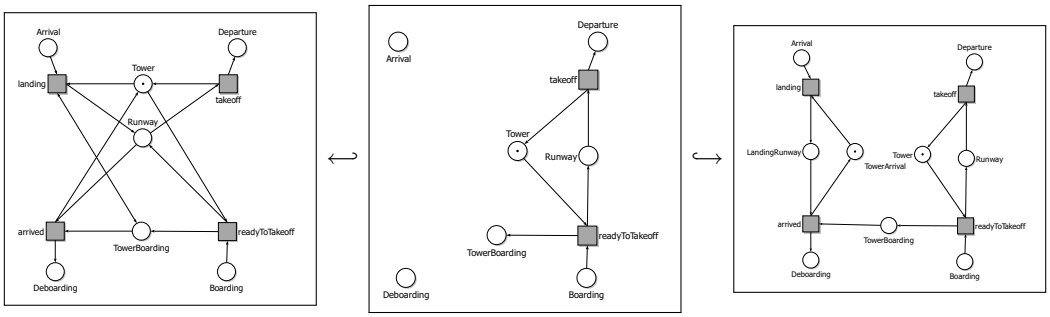
This negative application condition forbids the existence of an additional landing runway.

Rule 4b - NAC 2:



This negative application condition forbids the existence of an additional starting runway.

Rule 5a: *addSeperateLandingRunway*



By applying this rule to a level-1-airport, a seperate landing runway is added and the shared runway of the airport is automatically changed into a starting runway.

Rule 5b: *addLandingRunway*

This rule equals rule 3b with inverted edges up to isomorphism without negative application conditions.

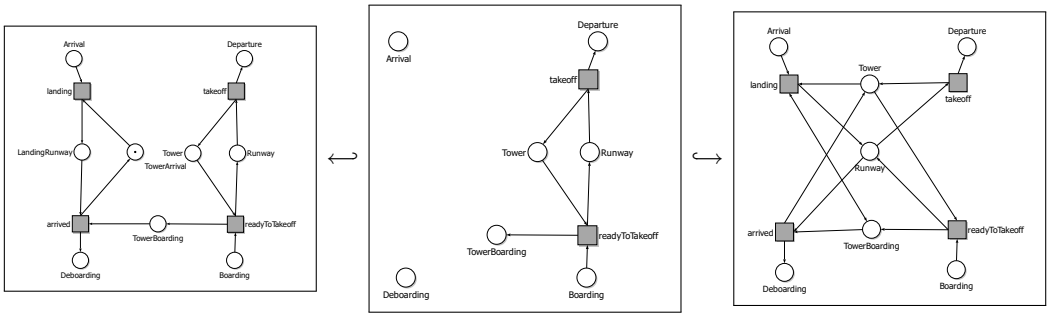
Adding an additional landing runway to a level-2-airport is modeled by this rule. It does not need a negative application condition because the transition *landing* of a level-1-airport has different pre- and post-domains than the transition *landing* of a level-2-airport. This fact prevents this rule from being applied to a level-1-airport.

Rule 6a: *removeLandingRunway*

This rule equals rule 4a with inverted edges up to isomorphism.

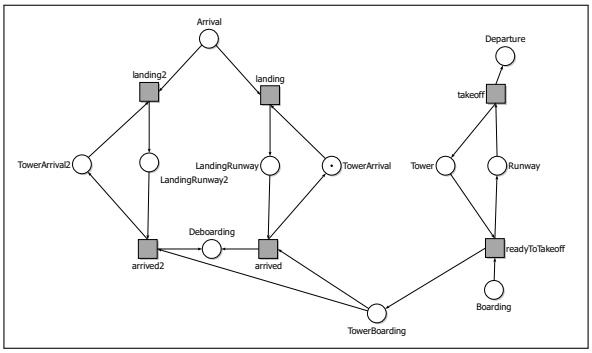
It describes the removing of an empty landing runway of a level-2-airport with multiple landing runways.

Rule 6b: *removeLastLandingRunway*



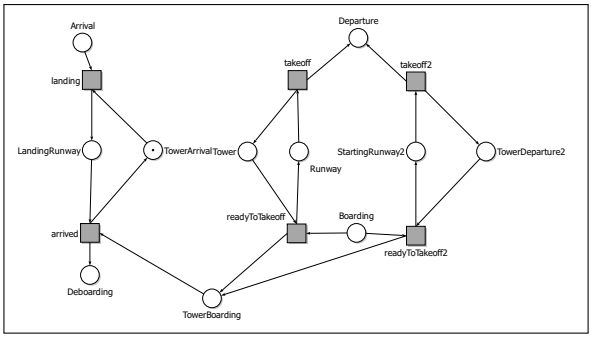
The removing of the last landing runway of a level-2-airport under the assumption that this landing runway is empty and only one starting runway exists is expressed by this rule.

Rule 6b - NAC 1:



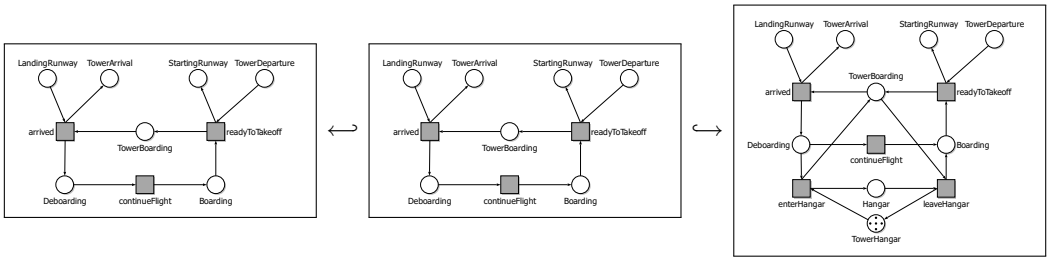
This negative application condition forbids the existence of an additional landing runway.

Rule 6b - NAC 2:



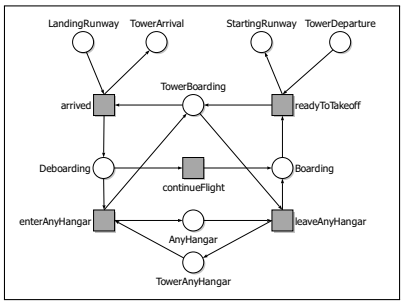
This negative application condition forbids the existence of an additional starting runway.

Rule 7a: $addHangarWithoutExistingMaintenance$



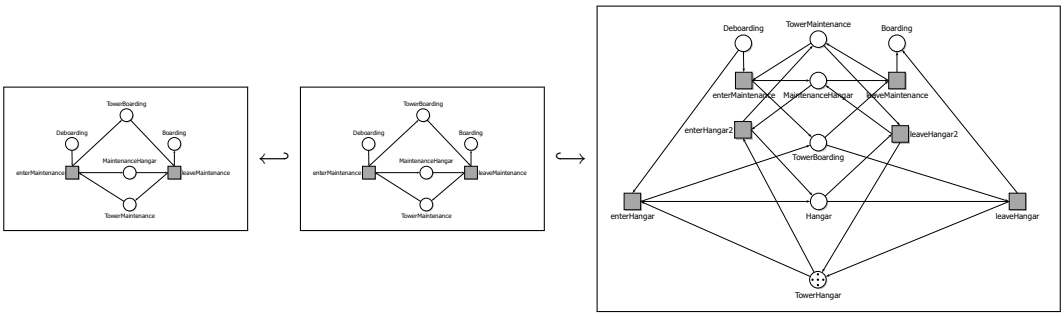
This rule adds a hangar, which provides space for up to five airplanes, to an airport that does not have a maintenance hangar yet. The left-hand side contains the transitions $arrived$ and $readyToTakeoff$ and the places $LandingRunway$, $TowerArrival$, $StartingRunway$ and $TowerDeparture$ only to prevent nonsensical matches.

Rule 7a - NAC 1:



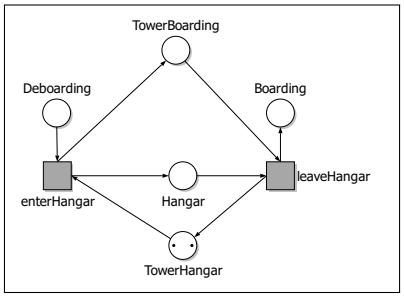
NAC 1 prevents this rule from being applied repeatedly. Additionally, if the airport already has a maintenance hangar, a mapping from this NAC into the airport net can be found and this rule cannot be applied.

Rule 7b: $addHangarWithExistingMaintenance$



If an airport already has a maintenance hangar, this rule can be applied. A hangar is added to the airport. It is reachable directly from the deboarding area and the maintenance hangar. Leaving the hangar, the airplanes can go directly to the boarding area or, alternatively, to the maintenance hangar. NACs are required to distinguish between a maintenance hangar and a hangar of an airport because they only differ in token count. Moreover, they have to prevent this rule from being applied repeatedly.

Rule 7b - NAC 1:



The morphism from the left-hand side L of this rule to this negative application condition NAC_1 is obvious but not an inclusion. Therefore, it is defined explicitly:

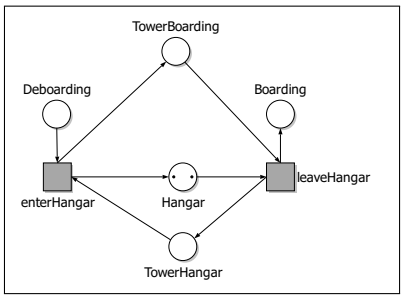
$$n_1 = (n_{1_P}, n_{1_T}) : L \to NAC_1 \text{ with}$$

$$\begin{aligned} n_{1_P} : L_P &\to NAC_{1_P} \text{ with} \\ TowerBoarding &\mapsto TowerBoarding \\ Deboarding &\mapsto Deboarding \\ Boarding &\mapsto Boarding \\ MaintenanceHangar &\mapsto Hangar \\ TowerMaintenance &\mapsto TowerHangar \end{aligned}$$

$$\begin{aligned} n_{1_T} : L_T &\to NAC_{1_T} \text{ with} \\ enterMaintenance &\mapsto enterHangar \\ leaveMaintenance &\mapsto leaveHangar \end{aligned}$$

For a more detailed descripion see NAC 2 of this rule.

Rule 7b - NAC 2:



The morphism from the left-hand side L of this rule to this negative application condition NAC_2 is obvious, but again not an inclusion. Therefore, it is defined explicitly as well:

$$\begin{aligned} n_2 : L &\to NAC_2 \text{ with} \\ n_2(x) &= n_1(x) \end{aligned}$$

This morphism is well-defined since n_1 is a valid **PTSys**-morphism and the places and the transitions are the same in NAC 1 and NAC 2.

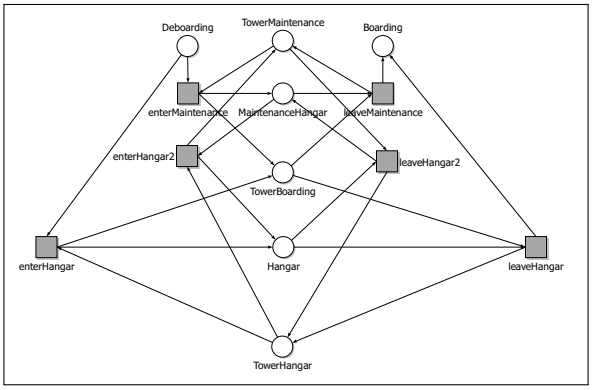
Negative application condition 1 and negative application condition 2 prevent a match $m : L \to AP$, mapping the place $MaintenanceHangar$ to the place $Hangar$ of the airport net AP, from being applied. This is shown in the following.
Let AP be a valid airport net with a hangar and a maintenance hangar and let

$m : L \to AP$ be a match that maps the place $MaintenanceHangar$ to the place $Hangar$ of the airport AP (satisfying the gluing condition in Fact A.21). Note that the hangar offers space for up to five airplanes, i.e. the sum of the tokens at the places $Hangar$ and $TowerHangar$ is always five, and the maintenance hangar is only used exclusively, i.e. the sum of the tokens at the places $MaintenanceHangar$ and $TowerMaintenance$ is always one. According to this property, one of the following inclusions morphisms exist:

- $q_1 : NAC_1 \to AP \in \mathcal{Q}$ with $q_1 \circ n_1 = m$
- $q_2 : NAC_2 \to AP \in \mathcal{Q}$ with $q_2 \circ n_2 = m$

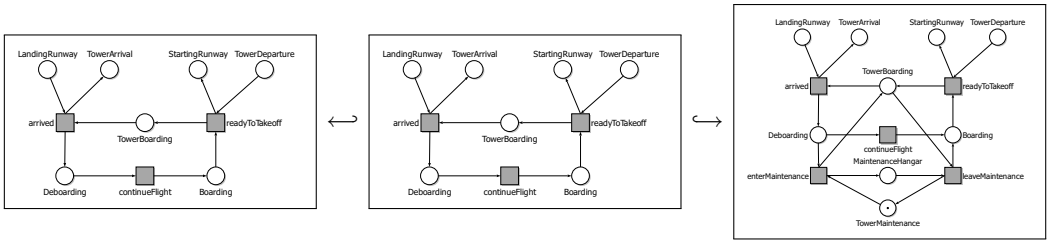
Therefore, the rule cannot be applied for m.
Obviously, if a match $m_2 : L \to AP_2$ maps the $MaintenanceHangar$ to the $MaintenanceHangar$ of AP, neither $q_3 : NAC_1 \to AP_2 \in \mathcal{Q}$ with $q_3 \circ n_1 = m_2$ nor $q_4 : NAC_1 \to AP_2 \in \mathcal{Q}$ with $q_4 \circ n_1 = m_2$ exist since the maintenance hangar only provides space for one airplane. So the rule can be applied (on the supposition that m_2 satisfies the gluing condition - Fact A.21 - and the following negative application condition NAC_3).

Rule 7b - NAC 3:



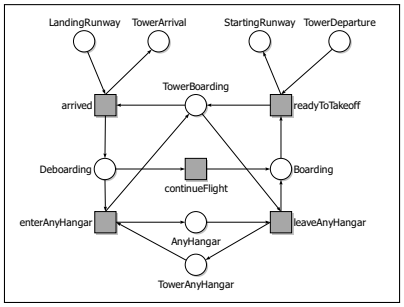
This NAC prevents this rule from being applied repeatedly.

Rule 8a: *addMaintenanceWithoutExistingHangar*

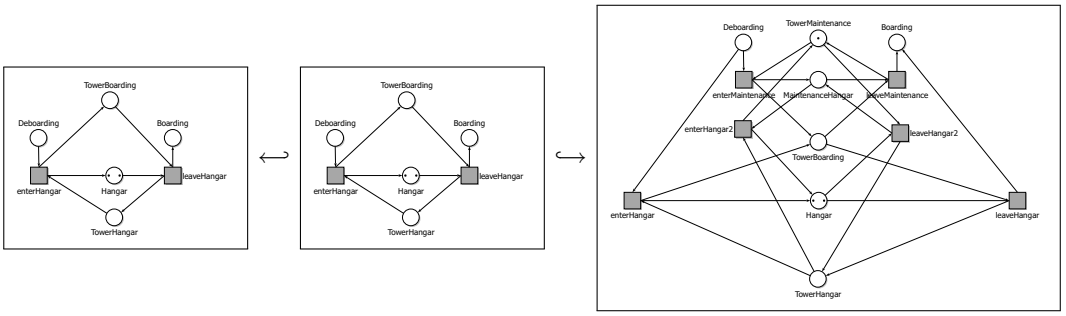


Analogous to rule 7a, except for building a maintenance hangar instead of a hangar.

Rule 8a - NAC 1:

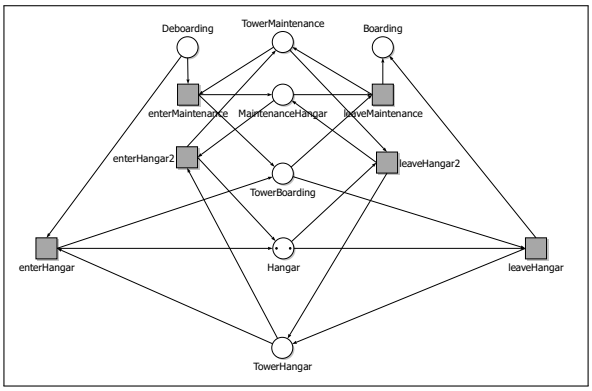


Rule 8b: $addMaintenanceWithExistingHangar1$



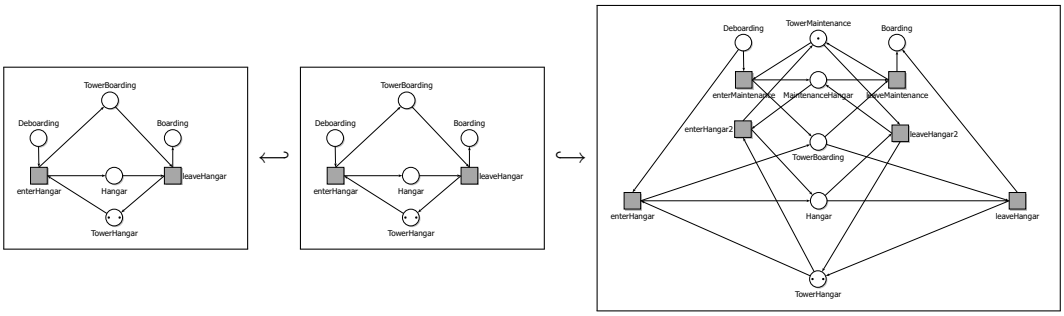
This and the following rule are required to model the adding of a maintenance hangar under the assumption that a hangar already exists. These rules resemble the previous rule. A mapping from the place $Hangar$ to the hangar of an airport and not to its maintenance hangar has to be ensured. Therefore, the property of the token count mentioned above is used again. The right-hand side of this rule corresponds to the right-hand side of the rule 7b.

Rule 8b - NAC 1:



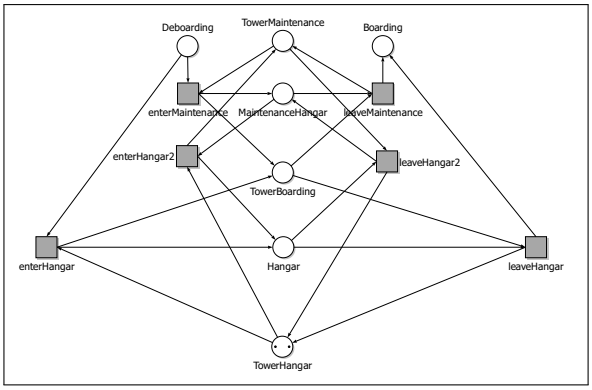
This NAC prevents this rule from being applied repeatedly.

Rule 8c: $addMaintenanceWithExistingHangar2$



This rule together with the previous rule are described above.

Rule 8c - NAC 1:



3.3.4 Remark for level-1-airports

Despite the existence of the additional edges between the transition *landing* and the place $TowerBoarding$, the situation that an arriving airplane cannot leave the runway and pass through the gate area can occur through the application of rules. One example for the development of this situation is that the last unused gate is removed while an airplane uses the shared runway of a level-1-airport for landing. This way the runway is blocked and no planes are able to takeoff because the arriving airplane cannot pass through the gate area and leave the runway.
It is possible to prevent this situation by using additional negative application conditions. Certainly, the example would be very voluminous and the additional negative application conditions would not provide any additional illustration in the context of this thesis. Thus, it is neglected in this particular example and a senseful application of rules is anticipated.

3.3.5 Applying Theoretical Results to ACS

In this section, it is illustrated how to apply some of the most important theoretical results to ACS. However, this is only realized exemplarily and not completely.

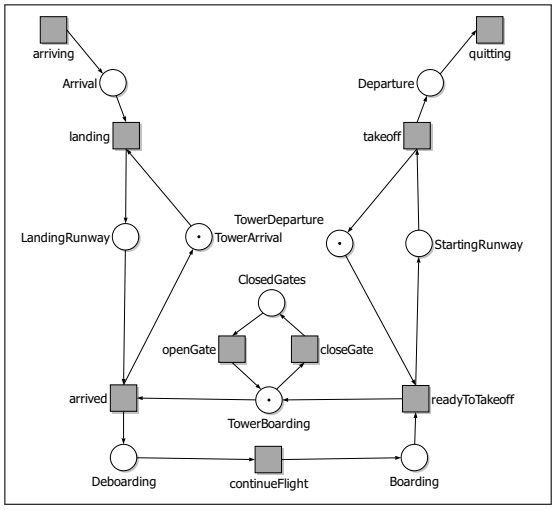


Figure 3.1: Level-2-airport G

The goal of this section is to show the practical relevance of the theoretical results introduced in Section 2.1 for this example.

Parallelism

Through parallelism, multiple independent changes of the airport in one step are possible. For example, a landing and a starting runway can be added to a level-2-airport at the same time. In the sequential case, the rules *addStartingRunway* and *addLandingRunway* are applied successively to the level-2-airport G shown in Figure 3.1. Obviously, these transformations are sequentially independent. Hence, it is possible to apply the *Local Church-Rosser Theorem with NACs* (see Theorem 2.31). It follows that the order of these transformations does not matter. Additionally, the *Parallelism Theorem with NACs* (see Theorem 2.33 and also 2.35) can be applied. This leads to the parallel rule (consisting of the componentwise disjoint unions of the original P/T systems) which can be used for performing both changes of the airport in one step. The parallel rule has one negative application condition consisting of the componentwise disjoint union of the NAC of rule *addStartingRunway* and the left-hand side of rule *addLandingRunway*.

Concurrency

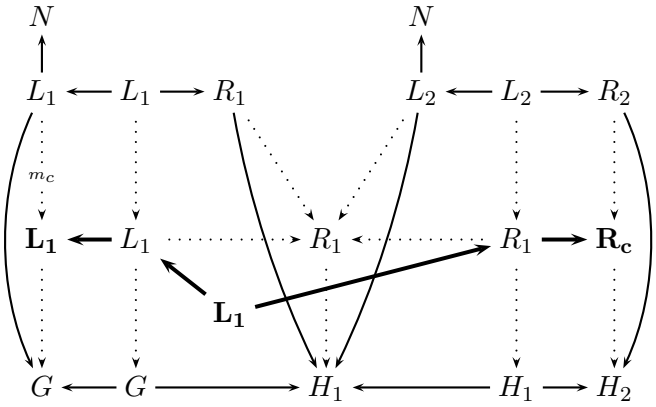
Successive addition of a hangar and a maintenance hangar to the level-2-airport G, depicted in Figure 3.1, are two sequentially dependent transformations. Thus, the Parallelism Theorem for performing both changes in one step cannot be applied. Nevertheless, the *Concurrency Theorem with NACs* (see Theorem 2.43) can

be applied and leads to the following diagram, where

$$p_1 = (L_1 \leftarrow L_1 \rightarrow R_1) := addHangarWithoutExistingMaintenance$$

and

$$p_2 = (L_2 \leftarrow L_2 \rightarrow R_2) := addMaintenanceWithExistingHangar2.$$



Note that this diagram is a special case since both rules are non-deleting.

The concurrent rule $p_c = (L_1 \leftarrow L_1 \rightarrow R_c)$ with NACs of the transformation sequence $G \overset{p_1}{\Rightarrow} H_1 \overset{p_2}{\Rightarrow} H_2$ is shown in Figure 3.2. The downward translation (see Definition 5.1 in [Lam07]) of the NAC of rule p_1 with respect to $m_c = id_{L_1} : L_1 \rightarrow L_1$ delivers an infinite set of negative application conditions, where the net part is identical to NAC 1 of rule p_1 and the marking of this net is arbitrary. Obviously, these NACs can be reduced to the one depicted in Figure 3.2. The down- and leftward translation (see Definition 5.6 in [Lam07]) of the NAC of rule p_2 delivers *true*.
This concurrent production is constructed according to Definition 2.41 and corresponds to the intuitive idea of a production creating a hangar and a maintenance hangar in one transformation step.

Note that the concurrent production of the transformation sequence, where the hangar is built after the maintenance hangar (this means application of rule $addMaintenanceWithoutExistingHangar$ and afterwards the application of rule $addHangarWithExistingMaintenance$), is identical up to isomorphism to the one presented here (with the same NAC).

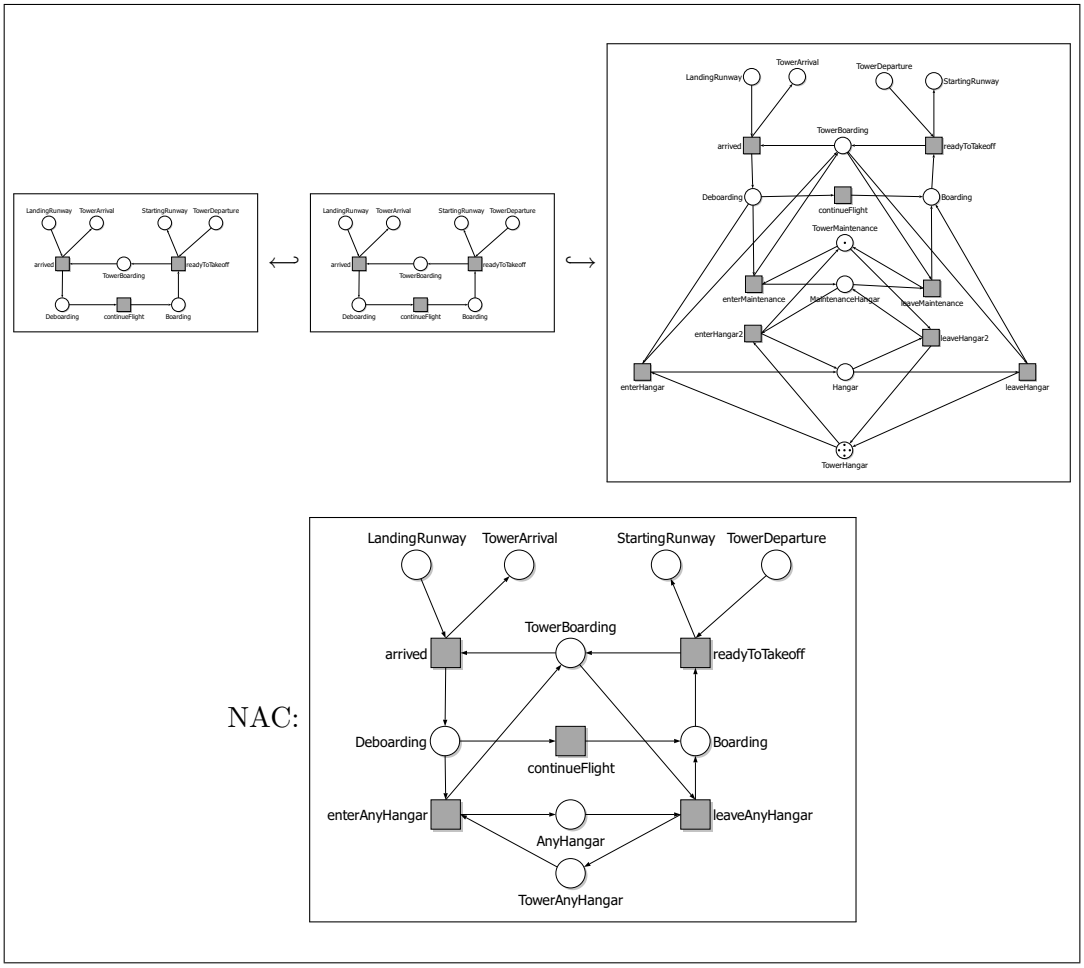


Figure 3.2: Concurrent Rule with NAC

Critical Pairs

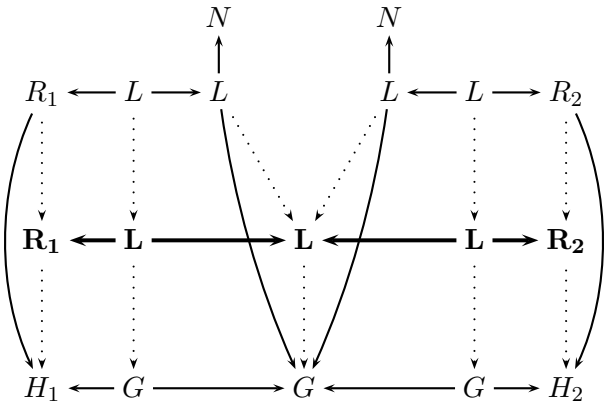
An example for parallel dependent transformations is the application of rules

$$p_1 = (L, L, R_1) := addMaintenanceWithoutExistingHangar$$

and

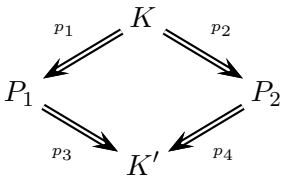
$$p_2 = (L, L, R_2) := addHangarWithoutExistingMaintenance$$

to the level-2-airport G, depicted in Figure 3.1. Note that this is a special case since the left-hand sides and the interfaces (K) of both rules are identical. So they are simply called L. This pair of transformations provokes two conflicts. On the one hand there is a produce-forbid-conflict and on the other hand there is a forbid-produce-conflict. *Completeness Theorem of Critical Pairs with NACs* (see Theorem 2.40) states the existence of a critical pair with extension diagrams. In this case, the critical pair corresponds to the rules themselves: $(K \overset{p_1}{\Rightarrow} P_1, K \overset{p_2}{\Rightarrow} P_2)$ with the negative application conditions of both rules, where $K = L$ is the left-hand side of rule p_1 (which equals the left-hand side of rule p_2) and $P_i = R_i$ is the right-hand side of rule p_i for $i = 1, 2$. Note that both rules have the same negative application condition N.



Confluence

In order to prove the local confluence of an adhesive HLR system, strict NAC-confluence (see Definition 2.48) of all critical pairs has to be shown, according to *Local Confluence Theorem with NACs* (see Theorem 2.49). Exemplarily, strict NAC-confluence of the critical pair defined above is proven.
First of all, the fact that the critical pair is confluent is shown.



This is obvious since rule

$$p_3 = (L_3, L_3, R_3) := addHangarWithExistingMaintenance$$

can be applied to P_1 and rule

$$p_4 = (L_4, L_4, R_4) := addMaintenanceWithExistingHangar2$$

can be applied to P_2. The result of these transformations is the same airport with a maintenance hangar and a hangar K' as pictured in Figure 3.3.

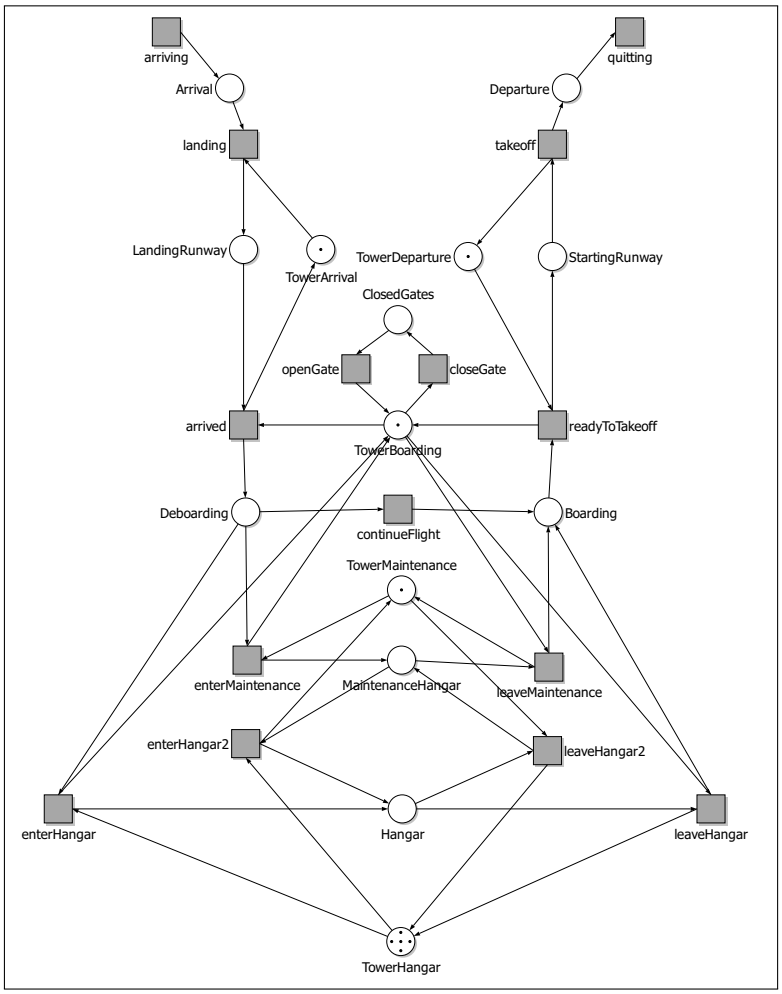
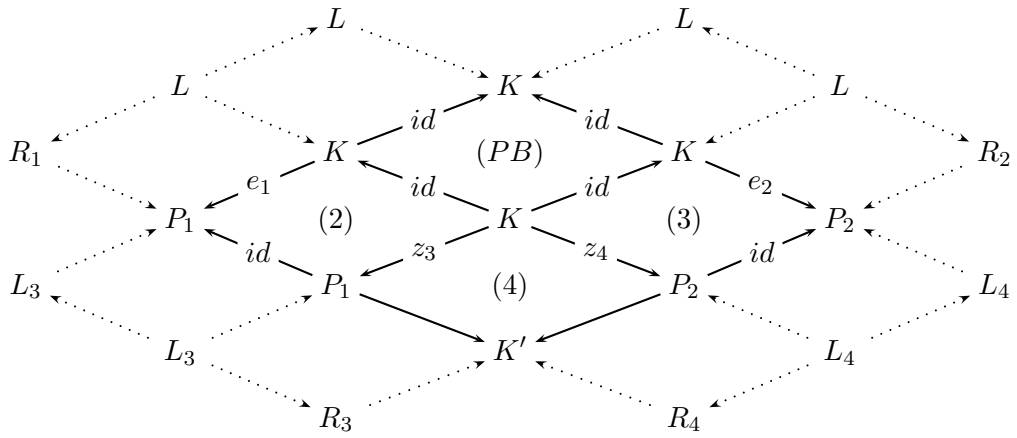


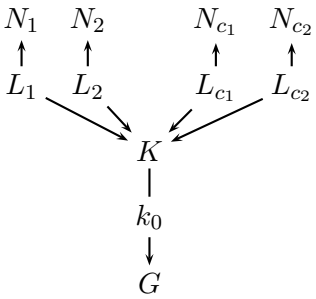
Figure 3.3: Result K' of transformation sequences $K \overset{p_1}{\Rightarrow} P_1 \overset{p_3}{\Rightarrow} K'$ and $K \overset{p_2}{\Rightarrow} P_2 \overset{p_4}{\Rightarrow} K'$

In the next step, the strictness of this critical pair is shown. Construct pullback

(PB) in the following diagram:

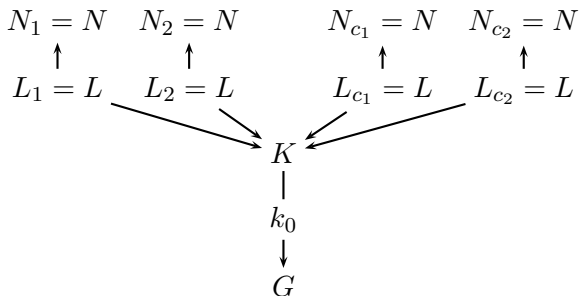


Morphisms $z_3 = e_1 : K \to P_1$ and $z_4 = e_2 : K \to P_2$ obviously exist such that (2), (3) and (4) commute. Intuitively, this is because the transformations $P_1 \overset{p_3}{\Rightarrow} K'$ and $P_2 \overset{p_4}{\Rightarrow} K'$ preserve all shared structures of $K \overset{p_1}{\Rightarrow} P_1$ and $K \overset{p_2}{\Rightarrow} P_2$.
Finally, NAC-confluence remains to be shown.



Therefore, the following statement has to be proven: Every morphism $k_0 : K \to G$ that is NAC-consistent (see Definition 2.44) with respect to $K \overset{p_1}{\Rightarrow} P_1$ and $K \overset{p_2}{\Rightarrow} P_2$ is also NAC-consistent with respect to $K \overset{p_1}{\Rightarrow} P_1 \overset{p_3}{\Rightarrow} K'$ and $K \overset{p_2}{\Rightarrow} P_2 \overset{p_4}{\Rightarrow} K'$.
For proving this property, it is necessary to construct the concurrent rules of both transformation sequences. This is already done above (see rule p_c in Figure 3.2 and remember that both concurrent rules are identical up to isomorphism). The negative application conditions as well as the left-hand sides of rules p_1 and p_2 are equal to

those of the concurrent rule(s) p_c (see Figure 3.2).



For this reason, strict NAC-confluence of this critical pair holds.

This result states that it does not matter whether the hangar or the maintenance hangar of an airport is built first because the results can be transformed to the same airport with a hangar and a maintenance hangar. More formally, all pairs of transformations where the critical pair can be embedded are locally confluent.

Tool Support

Since the construction of critical pairs and the verification of their strict NAC-confluence is very complex, a tool support is necessary. For showing the (local) confluence of a whole transformation system, all critical pairs have to be constructed and their strict NAC-confluence has to be shown. AGG (see [AGG08]) is a java-based tool developed by the *TFS-group* at the *Technische Universität Berlin* as environment for attributed graph transformation systems. AGG can handle negative application conditions and perform critical pair analyses of transformation systems. Note that AGG is still an ongoing project of the *TFS-group* and, therefore, still under development.

Chapter 4

Reconfigurable AHL Systems with NACs

4.1 Review of Reconfigurable AHL Systems

An AHL net is the extension of a P/T net by a data type. The data type consists of an algebraic specification and an algebra. This section contains a review of reconfigurable AHL systems with the most significant definitions and facts with respect to this thesis. In common literature, different definitions of AHL nets and AHL net morphisms can be found. The sets of variables of the specification and the sets of additional variables of the net part are neglected in most definitions even though they are used. Besides that, different types of AHL nets and AHL net morphisms exist. On the one hand there are so called *fixed AHL net morphisms* changing only the net part of an AHL net. On the other hand there are AHL net morphisms changing only the net structure and the algebra part of AHL nets. Generally, they are called *AHL net morphisms with fixed specification*. Finally, there are so called *generalized AHL net morphisms* changing also the specification and the algebra part of the data type. In this thesis, only the last called morphisms are considered and they are simply called *AHL net morphisms*.
To achieve the properties of a weak adhesive HLR category, these morphisms are restricted to those being isomorphisms on the algebra part. Additionally, a restriction to morphisms mapping the set of (specification) variables bijectively is reasonable because of the intuitive assumption that there are always enough variables available. Moreover, it is senseless to add or remove single variables by applying rules. Finally, AHL net morphisms are restricted to those mapping the additional (net) variables injectively in order to ensure preservation of firing.

Algebraic signatures, algebraic specifications and algebras are not introduced formally in this thesis. A short imposition to algebraic signatures and algebras can be found in Appendix B of [EEPT06]. A deeper introduction to this topic can be found in [EM85]. Note that in this thesis specification morphisms (**Spec**-morphisms) are

restricted to those mapping the sets of (specification) variables bijectively.

In the following subsection, AHL nets and the category of AHL nets and AHL net morphisms are introduced. Special morphisms, coproducts, pushouts and pullbacks of this category are defined in Appendix B.1 and their correctness is proven. The gluing condition for transforming AHL nets is defined and its correctness is proven in the appendix as well.
In the second subsection, AHL nets are expanded to AHL systems by adding markings. The category of AHL systems and morphisms is presented. For the sake of completeness, the most important concepts of this category, with respect to this thesis, are introduced in Appendix B.2 and their correctness is proven.

4.1.1 AHL Nets and the Category of AHL Nets

An AHL net can be considered as P/T net with a data type. The data type consists of an algebraic signature and an algebra. Every place has a determined type (sort) and can only contain tokens of this sort. Additionally, every transition has a set of fire conditions, i.e. equations which have to be fulfilled to enable the transition. The weight of the edges (i.e. the codomain of pre- and post-domain functions) are terms over the specification. The next definitions formalize AHL nets and AHL net morphisms.

Definition 4.1 (AHL Net)**.** An AHL net is given by

$$AN = (SP, P, T, pre, post, cond, type, A) \text{ with}$$

- algebraic specification $SP = (S, OP, E, X)$ with
  - sorts S
  - operations OP
  - equations E
  - additional variables X
- places P
- transitions T
- pre- and post-domain functions $pre, post : T \to (T_{OP}(X) \otimes P)^{\oplus}$, where $\otimes$ means the type-correct product
- fire conditions $cond : T \to \mathcal{P}_{fin}(Eqns(S, OP, X))$
- typing of places $type : P \to S$
- (S, OP, E)-algebra A

Note. In literature, the definitions of AHL nets differ. Usually, the sets of variables are neglected. Algebra A is often called SP-algebra, although the additional variables X are not used in the equations E of the specification. These variables are only used for the fire conditions and in arc descriptions (images of pre- and post-domain functions). Since the *type*-function was established later, it cannot be found in former literature.

As already mentioned, AHL net morphisms are restricted in this thesis. This restriction is formalized in the next definition.

Definition 4.2 ((Generalized) AHL Net Morphism (**AHLNet**-Morphism))**.** Given AHL nets $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2$, an AHL net morphism $f : AN_1 \to AN_2$ is given by

$$f = (f_{SP}, f_P, f_T, f_A) \text{ with}$$

- generalized algebra homomorphism (f_{SP}, f_A) with
  - specification morphism $f_{SP} = (f_S, f_{OP}, f_X) : SP_1 \to SP_2$ with a seperate injective mapping of the additional variables $f_X = (f_{X_s})_{s \in S_1} : X_1 \to X_2$
  - algebra isomorphism $f_A = (f_{A_s})_{s \in S_1} : A_1 \to V_{f_{SP}}(A_2)$
- $f_P : P_1 \to P_2$
- $f_T : T_1 \to T_2$

such that the following diagrams commute:

$$\begin{array}{ccccc}
\mathcal{P}_{fin}(Eqns(S_1, OP_1, X_1)) & \leftarrow cond_1 - & T_1 & \begin{array}{c} - pre_1 \rightarrow \\ - post_1 \rightarrow \end{array} & (T_{OP_1}(X_1) \otimes P_1)^{\oplus} \\
\downarrow & & \downarrow & & \downarrow \\
\mathcal{P}_{fin}(f_{SP}^{\#} \times f_{SP}^{\#}) & (=) & f_T & (=) & (f_{SP}^{\#} \otimes f_P)^{\oplus} \\
\downarrow & & \downarrow & & \downarrow \\
\mathcal{P}_{fin}(Eqns(S_2, OP_2, X_2)) & \leftarrow cond_2 - & T_2 & \begin{array}{c} - pre_2 \rightarrow \\ - post_2 \rightarrow \end{array} & (T_{OP_2}(X_2) \otimes P_2)^{\oplus}
\end{array}$$

$$\begin{array}{ccc}
P_1 & - type_1 \rightarrow & S_1 \\
\downarrow & & \downarrow \\
f_P & (=) & f_S \\
\downarrow & & \downarrow \\
P_2 & - type_2 \rightarrow & S_2
\end{array}$$

Remark 4.3 (AHL Net Morphisms).

1. $f_{SP}^{\#}$ is the extension of specification morphism f_{SP} to terms and equations (see Definition 8.2 in [EM85]) which maps the (specification) variables bijectively.

2. $V_{f_{SP}}$ is the forgetful functor with respect to f_{SP} (see Definition 8.1 (2.) in [EM85]).

3. $\mathcal{P}_{fin}$ is the power set functor that maps a set to its power set (see Example 7.9 in [EM85]).

4. f_X injective implies $Var(t) \subseteq Var(f_T(t))$ that ensures the preservation of firing.

Towards **AHLNet** as weak adhesive HLR category with NACs, some special morphism classes are defined in the following part.

Definition 4.4 (Injective, Surjective, (Jointly) Surjective and Bijective AHL Net Morphisms)**.** An **AHLNet**-morphism $f = (f_{SP}, f_P, f_T, f_A) : AN_1 \to AN_2$ as defined above is called injective (resp. surjective, bijective) if and only if f_P, f_T, f_{SP} and f_X are injective (resp. surjective, bijective).
A specification morphism $f_{SP} : SP_1 \to SP_2$ is called injective (resp. surjective, bijective) if and only if f is componentwise injective (resp. surjective, bijective).
A pair of **AHLNet**-morphisms $f = (f_{i_{SP}}, f_{i_P}, f_{i_T}, f_{i_A}) : AN_i \to AN_3$ for $i = 1, 2$ is called jointly surjective if and only if (f_{1_x}, f_{2_x}) are jointly surjective functions for $x = S, OP, X, P, T$.

Strict injectivity is a property of an AHL net morphism referring to the equations of the AHL nets. It states that no additional equations for mapped sorts and operations exist. The class of strict injective AHL net morphisms later forms morphism class $\mathcal{M}$.

Definition 4.5 (Strict Injective AHL Net Morphisms)**.** An injective **AHLNet**-morphism $f : AN_1 \to AN_2$ is called *strict injective* if and only if $f_{SP}^{\#^{-1}}(E_2) \subseteq E_1$, where f_{SP} is the specification morphism of f and E_i is the set of equations of AN_i for $i = 1, 2$.

A stronger property with respect to the equations than strict injectivity is *equation strictness* of AHL net morphisms. This property declares that the sets of equations of both AHL nets (the domain and the codomain) is exactly the same. Although, injectivity is not supposed. *Equation strict and surjective* morphisms later form morphism class $\mathcal{E}_0$ for $\mathcal{E}_0$-$\mathcal{M}$' factorization.

Definition 4.6 (Equation Strict AHL Net Morphisms)**.** An **AHLNet**-morphism $f : AN_1 \to AN_2$ is called *equation strict* if and only if $E_2 = f_{SP}^{\#}(E_1)$, where f_{SP} is the specification morphism of f and E_i are the sets of equations of AN_i for $i = 1, 2$.

Analogous to the case without data types, a class of jointly epimorphic morphisms is required for $\mathcal{E}'$-$\mathcal{M}'$ pair factorization (see Definition 2.6). Therefore, *jointly equation strict* morphisms are defined in the next step. This definition is similar to the definition of *equation strict* morphisms except for the fact that morphism pairs are considered.

Definition 4.7 (Jointly Equation Strict AHL Net Morphisms)**.** A pair of AHL net morphisms $f_1 : AN_1 \to AN_3$ and $f_2 : AN_2 \to AN_3$ with the same codomain is called *jointly equation strict* if and only if $E_3 = f_{1_{SP}}^{\#}(E_1) \cup f_{2_{SP}}^{\#}(E_2)$, where $f_{i_{SP}}$ are the specification morphisms of f_i for $i = 1, 2$ and E_i are the sets of equations of AN_i for $i = 1, 2, 3$.

Remark 4.8 (Strictness of AHL Net Morphisms). Note that the definitions of strictness of AHL net morphisms and P/T morphisms differ. The strictness of AHL net morphisms refers to the specification morphism as defined above. By contrast, the strictness of P/T morphisms refers to the marking of the places.
Equation strictness and injectivity of an AHL net morphism imply strict injectivity, although the inverse conclusion does not hold.

Towards the category **AHLNet**, a composition of AHL net morphisms is required. Since this composition is not as obvious as in the case without data types, it is formalized explicitly in the next definition.

Definition 4.9 (Composition of AHL Net Morphisms)**.** Given AHL nets $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2$.
The composition $(g \circ f)$ of **AHLNet**-morphisms $f = (f_{SP}, f_P, f_T, f_A) : AN_1 \to AN_2$ and $g = (g_{SP}, g_P, g_T, g_A) : AN_2 \to AN_3$ with $f_{SP} = (f_S, f_{OP}, f_X) : SP_1 \to SP_2$, $g_{SP} = (g_S, g_{OP}, g_X) : SP_2 \to SP_3$, $f_A : A_1 \to V_{f_{SP}}(A_2)$ and $g_A : A_2 \to V_{g_{SP}}(A_3)$ is given by

- the composition in **Sets** of places and transitions $g_P \circ f_P$ and $g_T \circ f_T$
- the composition of generalized algebra homomorphisms (f_{SP}, f_A) and (g_{SP}, g_A), i.e. $(g_{SP} \circ f_{SP}, V_{f_{SP}}(g_A) \circ f_A)$ with
  - the composition of specification morphisms $g_{SP} \circ f_{SP}$, i.e. the componentwise composition in **Sets** of sorts $g_S \circ f_S$, operations $g_{OP} \circ f_{OP}$ and additional variables $g_X \circ f_X$
  - the composition of algebra homomorphisms $V_{f_{SP}}(g_A) \circ f_A$, i.e. the componentwise composition in **Sets** $V_{f_{SP}}(g_A)_s \circ f_{A_s}$ for all sorts $s \in S_1$

Finally, the weak adhesive HLR category **AHLNet** is defined.

Fact 4.10 (Category **AHLNet** is a Weak Adhesive HLR Category)**.** AHL nets and **AHLNet**-morphisms as defined above form the category **AHLNet**, where the composition of morphisms is defined by the composition of **AHLNet**-morphisms as defined in Definition 4.9. This category is a weak adhesive HLR category (see Definition 2.3) with the special morphism class $\mathcal{M}$ of all *strict injective* (see Definition

4.5) **AHLNet**-morphisms (see Corollary 3.14 and Section 5 below Theorem 5.6 in [Pra08]).

4.1.2 AHL Systems and the Category of AHL Systems

An AHL system is an AHL net with an initial marking. In contrast to P/T systems, the marking of AHL systems consists of data types and not of indistinguishable tokens.

Definition 4.11 (AHL System)**.** An AHL system $AS = (AN, M)$ is an AHL net with an initial marking $M \in CP^{\oplus}$ with $CP = (A \otimes P) = \{(a, p) | a \in A_{type(p)}, p \in P\}$ for P being the places of AN. CP stands for *consistent place assignment*.

Definition 4.12 (AHL Morphism (**AHLSystems**-Morphism))**.** An AHL morphism $f : (AN_1, M_1) \to (AN_2, M_2)$ is given by an **AHLNet**-morphism $f = (f_{SP}, f_P, f_T, f_A)$ as defined in Definition 4.2 with the following additional property:

$$\forall (a, p) \in (A_1 \otimes P_1) : M_1(a, p) \leq M_2(f_A(a), f_P(p))$$

for A_1 being the algebra of AN_1 and P_1 being the places of AN_1.

Remark 4.13 (Restriction of AHL morphisms). In this thesis, only AHL morphisms with the mentioned restrictions of **AHLNet**-morphisms (see Definition 4.2) are used.

Definition 4.14 (Injective, (Jointly) Surjectivite, Bijectivite AHL Morphisms)**.** The definitions of injectivity, (jointly) surjectivity and bijectivity of AHL morphisms are equal to the definitions of the corresponding properties of AHL net morphisms (see Definition 4.4).

Analogous to the case without data types, some special morphism classes are required. On the one hand there are *marking strict AHL morphisms*, which are defined analogously to the case without data types (see Definition 3.10). On the other hand there are *minimal (jointly) surjective* AHL morphisms, which are also defined analogously to the case without data types (see Definitions 3.13 and 3.14). The second called morphism classes are called *minimal* since the marking of the target AHL system is minimal, i.e. every marking that is smaller that the minimal marking violates the well-definedness of the AHL morphism.

Definition 4.15 (Marking Strict AHL Morphism)**.** Given AHL systems $AS_i = (AN_i, M_i)$ for $i = 1, 2$, a *marking strict* AHL morphism $f = (f_{SP}, f_P, f_T, f_A) : AS_1 \to AS_2$ is an AHL morphism with the following additional property:

$$\forall (a, p) \in (A_1 \otimes P_1) : M_1(a, p) = M_2(f_A(a), f_P(p))$$

for A_1 being the algebra of AN_1 and P_1 being the places of AN_1.

Definition 4.16 (Strict AHL Morphism)**.** An AHL morphism is called *strict* if and only if it is *marking strict* (see Definition 4.15) and *strict injective* (see Definition 4.5).

Remark 4.17 (Strictness of AHL Morphisms). There are several definitions of strictness of AHL morphisms. On the one hand there is the marking strictness according to Definition 4.15. On the other hand there are two kinds of strictness with respect to the equations, the strict injectivity (see Definition 4.5) and the equation strictness (see Definition 4.6).
An AHL morphism is called *strict* if it is *marking strict* and *strict injective* (see Definition 4.16).

Fact 4.18 (Category **AHLSystems** is a Weak Adhesive HLR Category)**.** AHL systems and AHL morphisms form the category **AHLSystems**, where the composition of morphisms is defined componentwise as the composition of **AHLNet**-morphisms. This category is a weak adhesive HLR category (see Definition 2.3) with the special morphism class $\mathcal{M}_{strict}$ of all *strict*, i.e. *marking strict* (see Definition 4.15) and *strict injective* (see Definition 4.5), AHL morphisms (see Theorem 5.6 in [Pra08]).

Definition 4.19 (Minimal Surjective AHL Morphism)**.** A surjective AHL morphism (see Definition 4.14) $f : AS_1 \to AS_2$ with $AS_i = (AN_i, M_i)$ for $i = 1, 2$ is called *minimal* if and only if M_2 is minimal, i.e. the following statement holds for all $(a_2, p_2) \in A_2 \otimes P_2$:

$$M_2(a_2, p_2) = max(\{M_1(a, p) | a \in f_A^{-1}(a_2) \wedge p \in f_P^{-1}(p_2)\})$$

where P_2 are the places and A_2 is the algebra of AS_2, f_A is the algebra homomorphism and f_P is the mapping of the places of f.

Definition 4.20 (Minimal Jointly Surjective AHL Morphism)**.** A pair of jointly surjective AHL morphisms (see Definition 4.14) $f_i : AS_i \to AS_3$ with $i = 1, 2$ is called *minimal* if and only if M_3 is minimal, i.e. the following statement holds for all $(a_3, p_3) \in A_3 \otimes P_3$:

$$\begin{aligned} M_3(a_3, p_3) \quad = max(\quad & \{M_1(a, p) | a \in f_{1_A}^{-1}(a_3) \wedge p \in f_{1_P}^{-1}(p_3)\} \cup \\ & \{M_2(a, p) | a \in f_{2_A}^{-1}(a_3) \wedge p \in f_{2_P}^{-1}(p_3)\} \quad) \end{aligned}$$

where M_i is the marking, P_i are the places and A_i is the algebra of AS_i, f_{i_A} is the algebra homomorphism and f_{i_P} is the mapping of the places of f_i for $i = 1, 2$.

Since the category **AHLSystems** is a weak adhesive HLR category with the special morphism class of all *strict morphisms*, it is possible to formulate transformation rules in **AHLSystems** with two strict morphisms. Referring to the title of this thesis, the next definition introduces a reconfigurable AHL system.

Definition 4.21 (Reconfigurable AHL System)**.** Given an AHL system AS and a set $RULES$ of rules, a reconfigurable AHL system is defined by $(AS, RULES)$.

4.2 AHL Nets as weak Adhesive HLR Category with NACs

In this section is shown that the category **AHLNet** is a weak adhesive HLR category with NACs.

Theorem 4.22 ((**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) is a Weak Adhesive HLR Category with NACs)**.** *The category* (**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) *with the following morphism classes is a weak adhesive HLR category with NACs:*

$\mathcal{M}$	:	*strict injective AHL net morphisms (see Definition 4.5)*
$\mathcal{M}'$	:	*injective AHL net morphisms (see Definition 4.4 and Fact B.3)*
$\mathcal{Q}$	:	*injective AHL net morphisms (see Definition 4.4 and Fact B.3)*
$\mathcal{E}'$	:	*jointly equation strict and jointly surjective AHL net morphisms (see Definitions 4.7 and 4.4)*

Proof. As already mentioned, (**AHLNet**,$\mathcal{M}$) is a weak adhesive HLR category (see Fact 4.10). Therefore, merely the additional properties of a *weak adhesive HLR category with NACs* (see Definition 2.4) need to be proven. These properties are proven in the following Facts 4.23, 4.25, 4.26, 4.27, 4.28, 4.29, 4.30, 4.31, 4.32, 4.33 and 4.34. □

Remember that **AHLNet**-morphisms $f = (f_{SP}, f_P, f_T, f_A) : AN_1 \to AN_2$ with $f_{SP} = (f_S, f_{OP}, f_X)$ being a specification morphism and $f_A = (f_{A_s})_{s \in S}$ being an algebra homomorphism are restricted to AHL morphisms with the following properties, for AN_1 and AN_2 being AHL nets:

1. f_A is an isomorphism
2. f_X is injective
3. the mapping of the (specification) variables is implicitly bijective

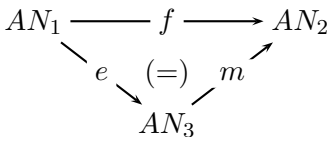
To simplify matters, (specification) variables are neglected in the following proofs. This is without without loss of generality because of the mentioned restrictions of **AHLNet**-morphisms. As well, the construction of the sets of variables is trivial where needed. Note that the (specification) variables are mapped implicitly by $f_{SP}^{\#}$. There is no explicit function mapping these variables. In contrast, the additional (net) variables X are mapped explicitly by the function f_X.

Fact 4.23 ((**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Unique Epi-$\mathcal{M}$ Factorization)**.** See Definition 2.5.

Proof. Given **AHLNet**-morphism $f : AN_1 \to AN_2$ with $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ and $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1..2$. An **AHLNet**-morphism is a tuple $f = (f_{SP}, f_P, f_T, f_A)$ with $f_{SP} = (f_S, f_{OP}, f_X)$ of functions (**Sets**-morphisms) and a unique epi-mono factorization exists for every function

(see Theorem 3.6.1 in [EMC+01]). In the following, epimorphism $e : AN_1 \to AN_3$ and monomorphism $m : AN_3 \to AN_2 \in \mathcal{M}$ are defined by componentwise epi-mono factorizations of f in **Sets** (with respect to places, transitions, sorts and operations of the signature and additional variables). The facts that $e = f$ and m is an inclusion hold. So an AHL net $AN_3 = (SP_3, P_3, T_3, pre_3, post_3, cond_3, type_3, A_3)$ with $SP_3 = (S_3, OP_3, E_3, X_3)$ can be constructed such that

- $S_3 = \{s \in S_2 | \exists s_1 \in S_1 : f_S(s_1) = s\}$
- $OP_3 = \{op \in OP_2 | \exists op_1 \in OP_1 : f_{OP}(op_1) = op\}$
- $E_3 = m_{SP}^{\#^{-1}}(E_2)$
- $X_{3_{e_S(s)}} = \{x \in X_{2_{f(s)}} | \exists x_1 \in X_{1_s} : f_X(x_1) = x\}$ for all sorts $s \in S_1$
- $A_3 = V_{m_{SP}}(A_2)$
- $P_3 = \{p \in P_2 | \exists p_1 \in P_1 : f_P(p_1) = p\}$
- $T_3 = \{t \in T_2 | \exists t_1 \in T_1 : f_T(t_1) = t\}$
- $pre_3 = pre_{2_{|T_3}}$
- $post_3 = post_{2_{|T_3}}$
- $type_3 = type_{2_{|P_3}}$
- $cond_3 = cond_{2_{|T_3}}$



Note that the proof for the net part is analogous to the proof of Fact 3.17. Well-definedness of $\Sigma_3 = (S_3, OP_3)$ follows directly from the well-definedness of epi-mono factorization in **Sets**. Additionally, the facts that E_3 is a set of equations over S_3 and OP_3 and A_3 is a SP_3-algebra follow directly by construction. X_3 is well-defined since e is an epimorphism.
Next, the well-definedness of e is shown. Suppose $e_{SP}^{\#}(E_1) \not\subseteq E_3$. Then $\exists e_1 \in E_1 : e_{SP}^{\#}(e_1) \notin E_3 \subseteq E_2$. Since $e_S = f_S$ and $e_{OP} = f_{OP}$, this is equivalent to $f_{SP}^{\#}(e_1) \notin E_3 \subseteq E_2$. From the well-definedness of f follows that $f_{SP}^{\#}(e_1) \in E_2 \backslash E_3$. The definition of E_3 implies that e_1 contains at least one sort or one operation that has no preimage in SP_3. By definition of S_3 and OP_3, this mentioned sort or operation of SP_3 has no preimage in SP_1, which is a contradiction to the assumption

that $e_1 \in E_1$. Therefore, $e^{\#}_{SP}(E_1) = f^{\#}_{SP}(E_1) \subseteq E_3$.
Next, the well-definedness of **AHLNet**-morphism e is shown.

$$\begin{aligned}(e^{\#}_{SP} \otimes e_P)^{\oplus} \circ pre_1 &= (f^{\#}_{SP} \otimes f_P)^{\oplus} \circ pre_1 \\ &= pre_2 \circ f_T \\ &= pre_3 \circ e_T\end{aligned}$$

post analogous.

$$\begin{aligned}e_S \circ type_1 &= f_S \circ type_1 \\ &= type_2 \circ f_P \\ &= type_3 \circ e_P\end{aligned}$$

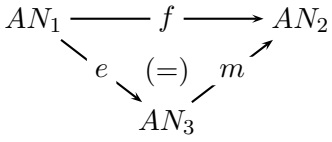
$$\begin{aligned}\mathcal{P}_{fin}(e^{\#}_{SP}) \circ cond_1 &= \mathcal{P}_{fin}(f^{\#}_{SP}) \circ cond_1 \\ &= cond_2 \circ f_T \\ &= cond_3 \circ e_T\end{aligned}$$

m is well-defined since it is an inclusion. Strictness of m holds by construction. Commutativity of $m \circ e = f$ follows from componentwise epi-mono factorizations in **Sets**.
Finally, the universal property is shown. Let $e' : AN_1 \to AN_4$ and $m' : AN_4 \to AN_2 \in \mathcal{M}$ with $m' \circ e' = f$ be **AHLNet**-morphisms, where e' is an epimorphism and $AN_4 = (SP_4, P_4, T_4, pre_4, post_4, cond_4, type_4, A_4)$ with $SP_4 = (S_4, OP_4, E_4, X_4)$. Unique isomorphisms $i_S : S_3 \to S_4$, $i_{OP} : OP_3 \to OP_4$ and $i_X = (i_{X_s})_{s \in S_3} : X_3 \to X_4$ with $i_x \circ e_x = e'_x \wedge m'_x \circ i_x = m_x$ for $x = S, OP, X$ are induced by epi-mono factorization in **Sets**. The fact that $i_\Sigma = (i_S, i_{OP})$ is a well-defined signature (iso)morphism is trivial to be shown. Equality of the sets of equations E_3 and E_4 up to isomorpism follows directly since $m, m' \in \mathcal{M}$ are strict injective and $i_\Sigma = (i_S, i_{OP})$ is an isomorphism with $m'_\Sigma \circ i_\Sigma = m_\Sigma$. So $i_{SP} = (i_S, i_{OP}, i_X)$ is a well-defined specification (iso)morphism.
The proof for the existence of unique isomorphisms i_x with $i_x \circ e_x = e'_x \wedge m'_x \circ i_x = m_x$ for $x = P, T$ is analogous to the corresponding proof without data types (see Fact 3.17). The existence of i_A is obvious since algebra homomorphisms are restricted to isomorphisms. Well-definedness of **AHLNet**-morphism $i = (i_{SP}, i_P, i_T, i_A)$ follows directly from the well-definedness of e, m, e' and m' and the mentioned commutativity.
Uniqueness of $i = (i_{SP}, i_P, i_T, i_A)$ is implied by the uniqueness of its components in **Sets**. □

Analogous to the proof that **PTSys** is a weak adhesive HLR category with NACs and according to Remark 5.26 in [EEPT06] (with $\mathcal{M}_0 = \mathcal{M}'$), the next fact shows that $(\mathbf{AHLNet}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has unique $\mathcal{E}_0$-$\mathcal{M}'$ factorization. This and the fact that **AHLNet** has binary coproducts (see Fact B.15) are used in the proof that **AHLNet** has unique $\mathcal{E}'$-$\mathcal{M}'$ pair factorization.

Fact 4.24 ((**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Unique $\mathcal{E}_0$-$\mathcal{M}'$ Factorization)**.** Let $\mathcal{E}_0$ be the class of equation strict (see Definition 4.6) and surjective AHL net morphisms. For every AHL net morphism $f : AN_1 \to AN_2$ a unique $\mathcal{E}_0$-$\mathcal{M}'$ factorization (AN_3, e, m) with $m \circ e = f$ exists, where $e \in \mathcal{E}_0$ and $m \in \mathcal{M}'$ with the following universal property:
For all AHL nets AN_3' and morphisms $e' \in \mathcal{E}_0 : AN_1 \to AN_3'$ and $m' \in \mathcal{M}' : AN_3' \to AN_2$ with $m' \circ e' = f$, a unique isomorphism $i : AN_3 \to AN_3'$ with $i \circ e = e'$ and $m' \circ i = m$ exists.

Proof. Given AHL nets $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2$ and **AHLNet**-morphism $f = (f_{SP}, f_P, f_T, f_A) : AN_1 \to AN_2$ with $f_{SP} = (f_S, f_{OP}, f_X)$. Construct AHL net $AN_3 = (SP_3, P_3, T_3, pre_3, post_3, cond_3, type_3, A_3)$ with $SP_3 = (S_3, OP_3, E_3, X_3)$ and morphisms $e = (e_{SP}, e_P, e_T, e_A) \in \mathcal{E}_0 : AN_1 \to AN_3$ with $e_{SP} = (e_S, e_{OP}, e_X)$ and $m = (m_{SP}, m_P, m_T, m_A) \in \mathcal{M}' : AN_3 \to AN_2$ with $m_{SP} = (m_S, m_{OP}, m_X)$ as described in Fact 4.23 except for the set of equations $E_3 = e_{SP}^{\#}(E_1)$.



Well-definedness of this construction in **AHLNet**, except for the new set of equations E_3, is already shown in the proof of Fact 4.23. Therefore, the fact that e_{SP} and m_{SP} are well-defined specification morphisms is shown in the next step. This follows directly by construction for morphism e_{SP}.
Note that morphism m is an inclusion and $m \circ e = f$ holds.

$$\forall eq_3 \in E_3 : \exists eq_1 \in E_1 : m_{SP}^{\#}(eq_3) = m_{SP}^{\#}(e_{SP}^{\#}(e_1)) = f_{SP}^{\#}(e_1) \in E_2$$

for any $eq_1 \in E_1$ with $e_{SP}^{\#}(eq_1) = eq_3$.
$e \in \mathcal{E}_0$ and $m \in \mathcal{M}'$ follow directly by construction.
Finally, the universal property is shown. Let $AN_3' = (SP_3', P_3', T_3', pre_3', post_3', cond_3', type_3', A_3')$ with $SP_3' = (S_3', OP_3', E_3', X_3')$ be an AHL net and $e' = (e'_{SP}, e'_P, e'_T, e'_A) \in \mathcal{E}_0 : AN_1 \to AN_3'$ with $e'_{SP} = (e'_S, e'_{OP}, e'_X)$ and $m' = (m'_{SP}, m'_P, m'_T, m'_A) \in \mathcal{M}' : AN_3' \to AN_2$ with $m'_{SP} = (m'_S, m'_{OP}, m'_X)$ be AHL net morphisms with $m' \circ e' = f$. The unique isomorphism $i = (i_{SP}, i_P, i_T, i_A) : AN_3 \to AN_3'$ with $i_{SP} = (i_S, i_{OP}, i_X)$ and $i \circ e = e'$ and $m' \circ i = m$ is induced as already shown in the proof of epi-$\mathcal{M}$-factorization (see Fact 4.23). However, $E_3 \cong E_3'$ remains to be shown since the set of equations E_3 differs from the corresponding set in the mentioned fact.
Assume $E_3 \ncong E_3'$. Then two cases can be distinguished:

1. $\exists eq_3 \in E_3 : \nexists eq_3' \in E_3' : i_{SP}^{\#}(eq_3) = eq_3'$. By construction, a set of preimages of eq_3 defined as $E_{eq_3} := \{eq \in E_1 | e_{SP}^{\#}(eq) = eq_3\} \subseteq E_1$ exists. The

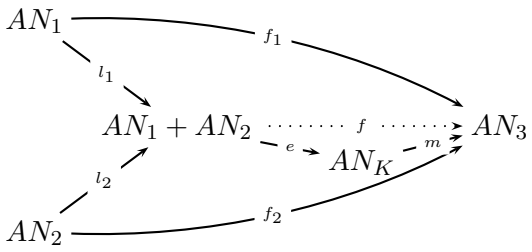
well-definedness of e' implies that $e'^{\#}_{SP}(E_1) \subseteq E_3$. Remember the fact that $i_\Sigma = (i_S, i_{OP})$ is an isomorphism is already shown in the proof of epi-$\mathcal{M}$-factorization (see Fact 4.23). $i_\Sigma \circ e_\Sigma = e'_\Sigma$ leads to the fact that all equations $eq_1 \in E_{eq_3}$ are mapped to the same equation by $e'^{\#}_{SP}$, i.e. $\exists eq'_3 \in E'_3 : \forall eq_1 \in E_{eq_3} : e'^{\#}_{SP}(eq_1) = eq'_3$ and $i^{\#}_{SP}(eq_3) = eq'_3$. This is a contradiction to the assumption that $\exists eq_3 \in E_3 : \nexists eq'_3 \in E'_3 : i^{\#}_{SP}(eq_3) = eq'_3$.

2. $\exists eq'_3 \in E'_3 : \nexists eq_3 \in E_3 : i^{\#}_{SP}(eq_3) = eq'_3$. Analogous, since this situation is symmetric.

□

Fact 4.25 (($\mathbf{AHLNet}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Unique $\mathcal{E}'$-$\mathcal{M}'$ Pair Factorization)**.** See Definition 2.6.

Construction. Let $f_1 : AN_1 \to AN_3$ and $f_2 : AN_2 \to AN_3$ with $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ and $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2, 3$ be AHL net morphisms. Since **AHLNet** has binary coproducts (see Fact B.15), there exist inclusions $l_1 : AN_1 \to AN_1 + AN_2$ and $l_2 : AN_2 \to AN_1 + AN_2$ and an induced morphism $f = [f_1, f_2] : AN_1 + AN_2 \to AN_3$ such that $f \circ l_1 = f_1$ and $f \circ l_2 = f_2$, where $AN_1 + AN_2$ is the binary coproduct of AN_1 and AN_2.
Let (AN_K, e, m) (with epimorphism $e \in \mathcal{E}_0$ and $m \in \mathcal{M}'$) be the $\mathcal{E}_0$-$\mathcal{M}$ factorization (see Fact 4.24) of f. $\mathcal{E}'$-$\mathcal{M}'$ pair factorization is defined by $(AN_K, (e_1 = e \circ l_1 : AN_1 \to AN_K, e_2 = e \circ l_2 : AN_2 \to AN_K), m : AN_K \to AN_3)$ with $(e_1, e_2) \in \mathcal{E}'$ and $m \in \mathcal{M}'$.



Proof.

1. $m \circ e_1 = m \circ e \circ l_1 = f \circ l_1 = f_1$.

2. $m \circ e_2 = m \circ e \circ l_2 = f \circ l_2 = f_2$.

3. $(e_1, e_2) \in \mathcal{E}'$. (l_1, l_2) are jointly surjective since they are coproduct inclusions (see Fact B.15) and the equations E_{1+2} of the coproduct object are defined as $E_{1+2} = E_1 \uplus E_2$.
   The fact that (e_1, e_2) are jointly epimorphic (jointly surjective) follows directly

from the composition property of epimorphisms and jointly epimorphic morphisms (see Fact D.1). $(e_1, e_2) \in \mathcal{E}'$ (are jointly equation strict) since $e \in \mathcal{E}_0$ (is equation strict) and $E_{1+2} = E_1 \uplus E_2$.

4. $m \in \mathcal{M}'$ follows directly from $\mathcal{E}_0$-$\mathcal{M}'$ factorization.

5. Uniqueness up to isomorphism of $\mathcal{E}'$-$\mathcal{M}'$ pair factorization. Let AN'_K be an AHL net and $e'_1 : AN_1 \to AN'_K$, $e'_2 : AN_2 \to AN'_K$ and $m' \in \mathcal{M}' : AN'_K \to AN_3$, where $(e'_1, e'_2) \in \mathcal{E}'$ and $m' \circ e'_1 = f_1$ and $m' \circ e'_2 = f_2$. Then an induced morphism $[e'_1, e'_2] : AN_1 + AN_2 \to AN'_K$ with $[e'_1, e'_2] \circ l_1 = e'_1$ and $[e'_1, e'_2] \circ l_2 = e'_2$ exists.
   In the following, the fact that $(AN'_K, [e'_1, e'_2], m')$ is the $\mathcal{E}_0$-$\mathcal{M}'$ factorization of f is shown.

$$f \circ l_i = f_i = m' \circ e'_i = m' \circ [e'_1, e'_2] \circ l_i \text{ for } i = 1, 2 \Rightarrow f = m' \circ [e'_1, e'_2]$$

   Since (e_1, e_2) are jointly surjective by assumption, $[e'_1, e'_2]$ is surjective. $E_{1+2} = E_1 \uplus E_2$ and the fact that $(e'_1, e'_2) \in \mathcal{E}'$ (are jointly equation strict) lead to $[e'_1, e'_2] \in \mathcal{E}_0$ (is equation strict). $m' \in \mathcal{M}'$ holds by assumption. This implies that $(AN'_K, [e'_1, e'_2], m')$ is the $\mathcal{E}_0$-$\mathcal{M}'$ factorization of f (see Fact 4.24). Uniqueness of this factorization implies the existence of a unique isomorphism $i : AN_K \to AN'_K$ with $i \circ e = [e'_1, e'_2]$ and $m' \circ i = m$. The fact that $i \circ e_j = e'_j$ for $j = 1, 2$ remains to be shown.

$$i \circ e_1 = i \circ e \circ l_1 = [e'_1, e'_2] \circ l_1 = e'_1$$
$$i \circ e_2 = i \circ e \circ l_2 = [e'_1, e'_2] \circ l_2 = e'_2$$

□

Fact 4.26 ((**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has $\mathcal{M}$-$\mathcal{M}'$ Pushout-Pullback Decomposition Property)**.** See Definition 2.7.

Proof. Given the following commutative diagram with $l \in \mathcal{M}$ and $w \in \mathcal{M}'$ in **AHLNet**, and where (1+2) is a pushout and (2) a pullback. Analogous to the first part of the proof of Fact 3.22, the facts that $v, z \in \mathcal{M}$ and (1+2) and (1) are pullbacks hold. Therefore, the fact that (1) and (2) are pushouts remains to be shown.

$$\begin{array}{ccccc} AN_0 & \xrightarrow{k} & AN_1 & \xrightarrow{r} & AN_2 \\ \downarrow l & (1) & \downarrow z & (2) & \downarrow v \\ AN_3 & \xrightarrow{u} & AN_4 & \xrightarrow{w} & AN_5 \end{array}$$

Let $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 0..5$ be AHL nets. The proof for the net part is analogous to the proof of Fact 3.22. So this property holds for the net part. Since $l_\Sigma = (l_S, l_{OP})$ and

$w_\Sigma = (w_S, w_{OP})$ are monomorphisms in **Sig**, which is an adhesive HLR category (see Fact 4.19 in [EEPT06]), $\mathcal{M}$ Pushout-Pullback Decomposition Lemma (see Theorem 4.26 (2.) in [EEPT06]) implies that (1) and (2) are pushouts in **Sig**. This lemma also leads to the fact that (1) and (2) are pushouts with respect to the additional variables X.
Next, $E_4 = z^\#_{SP}(E_1) \cup u^\#_{SP}(E_3)$ is shown. Suppose that $\exists e_4 \in E_4 \backslash (z^\#_{SP}(E_1) \cup u^\#_{SP}(E_3))$. Then $\exists e_5 \in E_5 : w^\#_{SP}(e_4) = e_5$. Since (1+2) is a pushout in **Spec**, $E_5 = v^\#_{SP}(E_2) \cup w^\#_{SP}(u^\#_{SP}(E_3))$ holds and two cases can be distinguished:

1. $e_5 \in w^\#_{SP}(u^\#_{SP}(E_3))$. So $\exists e_3 \in E_3, e'_4 \in E_4 : u^\#_{SP}(e_3) = e'_4 \wedge w^\#_{SP}(e'_4) = e_5$. The facts that w is injective and $w^\#_{SP}(e_4) = e_5$ imply that $e'_4 = e_4$, which is a contradiction to the assumption that $e_4 \in E_4 \backslash (z^\#_{SP}(E_1) \cup u^\#_{SP}(E_3))$.

2. $e_5 \in v^\#_{SP}(E_2)$. Pullback (2) implies that $E_1 = r^{\#^{-1}}_{SP}(E_2) \cap z^{\#^{-1}}_{SP}(E_4)$. Hence, $\exists e_1 \in E_1 : (v^\#_{SP}(r^\#_{SP}(e_1)) = e_5 \wedge z^\#_{SP}(e_1) = e_4)$. This is a contradiction to the assumption that $e_4 \in E_4 \backslash (z^\#_{SP}(E_1) \cup u^\#_{SP}(E_3))$.

Therefore, $SP_4 = SP_1 +_{SP_0} SP_3$ is a pushout in **Spec**. Obviously, $A_4 = A_1 +_{A_0} A_3$ is the amalgamated sum (see Lemma B.17) of A_1 and A_3 with respect to A_0 since algebra homomorphisms are restricted to isomorphisms. Hence, (1) is a pushout in **AHLNet**. Pushout decomposition implies that (2) is a pushout in **AHLNet**. □

Fact 4.27 ((**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has $\mathcal{M}$-$\mathcal{Q}$ Pushout-Pullback Decomposition Property)**.** See Definition 2.8. Analogous to Fact 4.26 since $\mathcal{Q}$=$\mathcal{M}'$.

Fact 4.28 ((**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Initial Pushouts over $\mathcal{M}'$-Morphisms)**.** See Definition 2.10.

Construction. Given an injective **AHLNet**-morphism $f : AN_0 \to AN_1 \in \mathcal{M}'$ with $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ and $SP_i = (S_i, OP_i, E_i)$ for $i = 0..1$, then the following dangling points are defined:

$$
\begin{aligned}
DP(P) \quad = \{ \quad & p \in P_0 \mid \exists t \in T_1 \backslash f_T(T_0) : \\
& pre_1(t) = \sum_{i=1}^{n} \lambda_i (r_i, p_i) \text{ with } \lambda_i \neq 0 \text{ and } f_P(p) = p_i \\
& \text{for some } i \text{ or} \\
& post_1(t) = \sum_{i=1}^{n} \lambda_i (r_i, p_i) \text{ with } \lambda_i \neq 0 \text{ and } f_P(p) = p_i \\
& \text{for some } i \}
\end{aligned}
$$

$$
DP(S) \quad = \quad DP'(S) \cup DP''(S) \cup DP'''(S) \text{ with}
$$

$$
\begin{aligned}
DP'(S) &= \{ \quad s \in S_0 \mid \exists op \in OP_1 \backslash f_{OP}(OP_0) \text{ which} \\
&\qquad \text{contains } f_S(s) \text{ as one of the sorts} \\
&\qquad \text{in its signature}\} \\
DP''(S) &= \{ \quad s \in S_0 \mid \exists e \in E_1 \backslash f^{\#}_{SP}(E_0) \text{ which} \\
&\qquad \text{contains } f_S(s) \text{ as one of the sorts} \\
&\qquad \text{in its signature}\} \\
DP'''(S) &= \{ \quad s \in S_0 \mid \exists p \in P_1 \backslash f_P(P_0) : type_1(p) = f_S(s)\}
\end{aligned}
$$

$$
\begin{aligned}
DP(T_{OP}) = \{ \quad & r \in T_{OP_0}(X_0) \mid \exists t \in T_1 \backslash f_T(T_0) : \\
& pre_1(t) = \sum_{i=1}^{n} \lambda_i (r_i, p_i) \text{ with } \lambda_i \neq 0 \text{ and } f^{\#}_{SP}(r) = r_i \\
& \text{for some } i \text{ or} \\
& post_1(t) = \sum_{i=1}^{n} \lambda_i (r_i, p_i) \text{ with } \lambda_i \neq 0 \text{ and } f^{\#}_{SP}(r) = r_i \\
& \text{for some } i\}
\end{aligned}
$$

$$
\begin{aligned}
DP(EQNS) = \{ \quad & e \in EQNS(\Sigma_0) \mid \exists t \in T_1 \backslash f_T(T_0) : f^{\#}_{SP}(e) \in cond_1(t)\}, \\
& \text{where } \Sigma_0 = (S_0, OP_0) \text{ is the signature of } AN_0
\end{aligned}
$$

$$
\begin{aligned}
DP(OP) = \{ \quad & op \in OP_0 \mid \exists e \in E_1 \backslash f^{\#}_{SP}(E_0) \text{ which contains } f_{OP}(op) \text{ as one} \\
& \text{of its operations}\}
\end{aligned}
$$

Note that $DP''(S)$ and $DP(OP)$ are required to satisfy the second part of the **Spec** gluing condition (see Fact B.25).

The boundary $AN_B = (SP_B, P_B, \emptyset, pre_B, post_B, cond_B, type_B, A_B)$ can be constructed as follows:

- $P_B = DP(P)$
- $T_B = \emptyset$
- $pre_B, post_B : \emptyset \to (T_{OP_B}(X) \otimes P)^{\oplus}$
- $SP_B = (S_B, OP_B, E_B)$ with
  - $S_B = DP(S) \cup type_0(DP(P)) \cup sorts(DP(T_{OP})) \cup sorts(DP(EQNS))$
  - $OP_B = DP(OP) \cup opns(DP(T_{OP})) \cup opns(DP(EQNS))$
  - $E_B = b^{\#^{-1}}_{SP}(E_0)$, where b_{SP} is the inclusion given by $S_B \subseteq S_0$ and $OP_B \subseteq OP_0$.
  - $X_{B_s} = \{x \in X_{0_s} | \exists t \in T_1 \backslash f_T(T_0) : \quad (cond_1(t) \text{ contains } f_{X_s}(x)) \vee (pre_1(t) \text{ contains } f_{X_s}(x)) \vee (post_1(t) \text{ contains } f_{X_s}(x))\}$ for all $s \in S_B \subseteq S_0$

where $sorts(TEQ)$ is the set of all sorts of TEQ and $opns(TEQ)$ is the set of all operations of TEQ for TEQ being a set of equations or a set of terms.

- $A_B = V_{b_{SP}}(A_0)$.
- $type_B = type_{0_{|P_B}}$
- $cond_B : \emptyset \to \mathcal{P}_{fin}(Eqns(\Sigma_B, X))$

The context object $AN_C = (SP_C, P_C, T_C, pre_C, post_C, cond_C, type_C, A_C)$ can be constructed as pushout complement in **AHLNet** (see part 2 of proof of Fact B.26):

- $P_C = (P_1 \backslash f_P(P_0)) \cup f_P(b_P(P_B))$
- $T_C = (T_1 \backslash f_T(T_0)) \cup f_T(b_T(T_B)) = (T_1 \backslash f_T(T_0)) \cup f_T(b_T(\emptyset)) = (T_1 \backslash f_T(T_0))$
- $pre_C = pre_{1_{|T_C}}$
- $post_C = post_{1_{|T_C}}$
- $SP_C = (S_C, OP_C, E_C)$ with
  - $S_C = (S_1 \backslash f_S(S_0)) \cup f_S(b_S(S_B))$
  - $OP_C = (OP_1 \backslash f_{OP}(OP_0)) \cup f_{OP}(b_{OP}(OP_B))$
  - $E_C = (E_1 \backslash f^{\#}_{SP}(E_0)) \cup f^{\#}_{SP}(b^{\#}_{SP}(E_B))$
  - $X_{C_{f'_S(s)}} = (X_{1_{f_S(s)}} \backslash f_{X_s}(X_{0_s})) \cup f_{X_s}(b_{X_s}(X_{B_s}))$ for all $s \in S_B \subseteq S_0$. Remember b is an inclusion, f_{X_s} is injective and $X_{B_s} \subseteq X_{0_s}$ for every $s \in S_B$.
  - $X_{C_s} = X_{1_s}$ for all $s \notin f_S(S_0)$
- $A_C = V_{c_{SP}}(A_1)$, where c_{SP} is the inclusion given by $S_C \subseteq S_1$ and $OP_C \subseteq OP_1$.
- $type_C = type_{1_{|P_C}}$
- $cond_C = cond_{1_{|T_C}}$

The monomorphisms $b : AN_B \to AN_0 \in \mathcal{M}$ and $c : AN_C \to AN_1 \in \mathcal{M}$ are inclusions and the monomorphism $f' : AN_B \to AN_C \in \mathcal{M}'$ can be constructed as restriction of f to AN_B: $f' = f_{|AN_B}$. This construction leads to *initial pushout* (1).

$$\begin{array}{ccc} AN_B & \xrightarrow{b} & AN_0 \\ \downarrow{\scriptstyle f'} & (1) & \downarrow{\scriptstyle f} \\ AN_C & \xrightarrow{c} & AN_1 \end{array} \qquad \begin{array}{ccccc} AN_B & \overset{b^*}{\dashrightarrow} & AN_2 & \xrightarrow{b'} & AN_0 \\ \downarrow{\scriptstyle f'} & (3) & \downarrow{\scriptstyle f''} & (2) & \downarrow{\scriptstyle f} \\ AN_C & \overset{c^*}{\dashrightarrow} & AN_3 & \xrightarrow{c'} & AN_1 \end{array} \quad \text{with } AN_B \xrightarrow{b} AN_0 \text{ and } AN_C \xrightarrow{c} AN_1$$

Proof. **Well-definedness of AN_B.** The proof for the well-definedness of the net part (the skeleton of the net) is analogous to the case without data types (see Fact 3.24). The construction implies that S_B contains all required sorts for the places in P_B. Since T_B is empty, no operations are required for the net part.
Next, the well-definedness of SP_B and A_B is shown. Let $op \in OP_B$ be an operation. By construction, all sorts of the signature of op are in the set S_B. $E_B = b^{\#^{-1}}(E_0)$ ensures that E_B only contains equations over the signature $\Sigma_B = (S_B, OP_B)$.
Obviously, A_B is well-defined.

Well-definedness and strict injectivity of b are obvious since b is an inclusion, $T_B = \emptyset$, $E_B = b^{\#^{-1}}(E_0)$ and $type_B = type_{0_{|P_B}}$.

Well-definedness of AN_C, morphisms c and f and pushout (1). Since AN_C is constructed as pushout complement, it is sufficient to show the satisfaction of the gluing condition for $AN_B \xrightarrow{b} AN_0 \xrightarrow{f} AN_1$ (see Fact B.26).
In the first step, the satisfaction of the **Spec** gluing condition is shown. Because of the injectivity of f, the set of identification points is always empty. $DP'(S) \subseteq b_S(S_B)$ follows directly by construction.
The satisfaction of the **Spec** gluing condition requires to show that the set E_C is a set of equations over the signature $\Sigma_C = (S_C, OP_C)$. Let $e_C \in E_C \subseteq E_1$ be an equation. Several cases can be distinguished:

1. $e_C \in f_{SP}^{\#}(b_{SP}^{\#}(E_B))$. For all sorts s and all operations op of e_C, the fact that $s \in f_S(b_S(S_B))$ and $op \in f_S(b_S(S_B))$ hold. By construction of AN_C, $s \in S_C$ and $op \in OP_C$ follow directly.

2. $e_C \in E_1 \backslash f_{SP}^{\#}(E_0)$. For all sorts s of e_C, several cases can be distinguished:

   (a) $s \in f_S(b_S(S_B))$. This means s has a preimage in S_B. By construction of AN_C, the fact that $s \in S_C$ holds.

   (b) $s \in f_S(S_0) \backslash f_S(b_S(S_B))$. Then $\exists s_0 \in S_0 \backslash b_S(S_B) : f_S(s_0) = s$ and, by definition, $s_0 \in DP(S)$. Note that $DP(S) \subseteq S_B$. This implies that $s \in f_S(b_S(S_B))$, which is a contradiction to the assumption that $s \in f_S(S_0) \backslash f_S(b_S(S_B))$.

   (c) $s \in S_1 \backslash f_S(S_0)$. The construction of Σ_C leads to $s \in S_C$.

   The proof for the operations of the equation e_C is analogous.

So the **Spec** gluing condition (see Fact B.25) holds.
The proof for the satisfaction of the dangling condition for the places is analogous to the corresponding proof without data types in **PTNet** (see Fact 3.24).
$DP(T_{OP}) \subseteq b_{SP}^{\#}(T_{OP_B}(X))$, $DP(EQNS) \subseteq b_{SP}^{\#}(T_{OP_B}(X))$ and $DP'''(S) \subseteq b_S(S)$ follow directly by construction of AN_B.
Since the gluing condition for $AN_B \xrightarrow{b} AN_0 \xrightarrow{f} AN_1$ is satisfied, the pushout complement AN_C and the morphisms f' and c are well-defined and (1) is a pushout (see

Fact B.26). Additionally, c is strict injective since (1) is a pushout.

Initiality of pushout (1) remains to be shown. Let (2) be a pushout with $b', c' \in \mathcal{M}$. Note that (2) satisfies the gluing condition (see Fact B.26).

Existence and well-definedness of $b^* \in \mathcal{M}$ and $c^* \in \mathcal{M}$ and required commutativity. The proof for the existence and well-definedness of the net part is analogous to the case without data types (see Fact 3.24).
In the first step, the existence of a signature morphism $b^*_\Sigma = (b^*_S, b^*_{OP})$ is shown. Define $b^*_\Sigma := b'^{-1}_\Sigma \circ b_\Sigma$. Since $S_B \subseteq S_0$ and $OP_B \subseteq OP_0$ only contain dangling points, it is sufficient for the well-definedness of b'^{-1}_Σ for $b_\Sigma(AN_B)$ to show that every dangling point has a preimage in S_2 resp. OP_2. Note that b' is strict injective. Let $s_B \in S_B$ be a sort of S_B. Several cases can be distinguished:

1. $s_B \in sorts(DP(T_{OP}))$. By gluing condition, $DP(T_{OP}) \subseteq b'^{\#}_{SP}(T_{OP_2}(X))$ holds, i.e. $\exists! s_2 \in S_2 : b'_S(s_2) = s_B$. So $b'^{-1}_S(s_B) = s_2$. Note that s_2 is unique since b' is injective.

2. $s_B \in sorts(DP(EQNS))$. Analogous.

3. $s_B \in type_0(DP(P))$. Analogous.

4. $s_B \in DP'(S)$. Analogous.

5. $s_B \in DP'''(S)$. Analogous.

6. $s_B \in DP''(S)$. Due to the gluing condition of $AN_2 \xrightarrow{b'} AN_0 \xrightarrow{f} AN_1$, $E_3 = E_1 \backslash f^{\#}_{SP}(E_0) \cup f^{\#}_{SP}(b'^{\#}_{SP}(E_2))$ is a set of equations over $\Sigma_3 = (S_3, OP_3)$. To show that this condition is sufficient for the existence of $s_2 \in S_2$ with $b'_S(s_2) = s_B$, assume $s_B \notin b'_S(S_2)$. Remember that $s_B \in DP''(S)$. The definition of $DP''(S)$ implies $\exists e_1 \in E_1 \backslash f^{\#}_{SP}(E_0)$ which contains $f_S(s_B)$ as one of the sorts in its signature. By gluing condition of (2), $E_3 = E_1 \backslash f^{\#}_{SP}(E_0) \cup f^{\#}_{SP}(b'^{\#}_{SP}(E_2))$ and $S_3 = S_1 \backslash f_S(S_0) \cup f_S(b'_S(S_2))$. So $\exists e_3 \in E_3 : c'^{\#}_{SP}(e_3) = e_1$ and $\nexists s_3 \in S_3 : c'_S(s_3) = f_S(s_B)$. Since $e_3 \in E_3$ contains $s_3 \notin S_3$ as one of the sorts in its signature, E_3 is no set of equations over Σ_3. The violation of the gluing condition is a contradiction to the fact that (2) is a pushout. Hence, $\exists! s_2 \in S_2 : b'_S(s_2) = s_B$. So $b'^{-1}_S(s_2) = s_B$. Note that s_2 is unique since b' is injective.

The proof for the operations and the definition of b_{OP} is analogous.
Well-definedness of b'^{-1}_Σ and well-definedness of b_Σ imply the well-definedness of b^*_Σ. By construction, $b'_\Sigma \circ b^*_\Sigma = b_\Sigma$ holds.
Next, the preservation of the equations of E_B by b^*_{SP} is shown. $b^{\#}_{SP}(E_B) \subseteq E_0$, $b'^{\#^{-1}}_{SP}(E_0) \subseteq E_2$ (strictness of b'_{SP}) and $b'_{SP} \circ b^*_{SP} = b_{SP}$ imply that every equation $e_B \in E_B \subseteq E_0$ can be translated to $b'^{\#^{-1}}_{SP}(e_B) = e_2 \in E_2$ with $b^{*\#}_{SP}(e_B) = e_2$. Therefore, $b^{*\#}_{SP}(E_B) \subseteq E_2$, i.e. b^*_{SP} is a well-defined specification morphism.

Define $b_X^* := b_X'^{-1} \circ b_X$. For the well-definedness of $b_X'^{-1}$ it is sufficient to show that every dangling point of X_0 has a preimage in X_2 since X_B only contains dangling points. This follows directly from the gluing condition of (2).
Since algebra homomorphisms are restricted to isomorphisms, b_A^* obviously exists (and is unique). $b_S^* \circ type_B = type_2 \circ b_P^*$ follows directly by construction. $\mathcal{P}_{fin}(b_{SP}^{*\#}) \circ cond_B = cond_2 \circ b_T^*$ and $(b_{SP}^{*\#} \otimes b_P^*)^{\oplus} \circ pre_B = pre_2 \circ b_T^*$ (*post* analogous) are obvious since T_B is empty. So b^* is a well-defined **AHLNet**-morphism and $b' \circ b^* = b$.
b^* is strict injective, i.e. $b^* \in \mathcal{M}$, since $\mathcal{M}$-morphisms are closed under decomposition in general (weak) adhesive HLR categories and $b' \circ b^* = b \in \mathcal{M}$ and $b' \in \mathcal{M}$.

In the next step, morphism c_{SP}^* is defined and the facts proven above for morphism b_{SP}^* are proven for morphism c^*.
Define $c_\Sigma^* := c_\Sigma'^{-1} \circ c_\Sigma$. To show the well-definedness of $c_\Sigma'^{-1}$ for $c_\Sigma(AN_C)$, it is sufficient to show that every sort (resp. operation) of S_C (resp. OP_C) has a preimage in S_3 (resp. OP_3). Let $s_C \in S_C$ be a sort of S_C. Remark that $S_C \subseteq S_1$. Two cases can be distinguished:

1. $s_C \in S_1 \backslash f_S(S_0)$. The construction of the pushout complement implies that $\exists! s_3 \in S_3 : c_S'(s_3) = s_C$. So $c_S'^{-1}(s_3) = s_C$. Note that s_3 is unique since c' is injective. (*)

2. $s_C \in f_S(b_S(S_B))$. As already shown, this is equivalent to $s_C \in f_S(b_S'(b_S^*(S_B)))$. This implies $s_C \in f_S(b_S'(S_2))$. Since (2) is a pushout, $\exists! s_2 \in S_2 : c_S'(f_S''(s_2)) = s_C$. Note that the uniqueness of s_2 follows from the injectivity of c_S'. Hence $c_S'^{-1}(s_C) = f_S''(s_2)$. (**)

The proof for the operations is analogous.
So c_Σ^* is a well-defined signature morphism with $c_\Sigma' \circ c_\Sigma^* = c_\Sigma$. Additionally, the injectivity of c_Σ^* follows from the injectivity of c_Σ and c_Σ'.
The proofs that c_{SP}^* is a well-defined specification morphism, i.e. $c_{SP}^{*\#}(E_C) \subseteq E_3$, and that c^* is strict injective, i.e. $c^* \in \mathcal{M}$, are analogous to the proofs for morphism b^*.
Define $c_X^* := c_X'^{-1} \circ c_X$. In the next step, the well-definedness of $c_X'^{-1}$ is shown. Therefore, it is sufficient to show that every variable x_C in $X_{C_{s_C}} \subseteq X_{1_{s_C}}$ has a preimage in $X_{3_{c_{SP}^*(s_C)}}$, for $s_C \in S_C \subseteq c_{SP}'(S_3) \subseteq S_1$ being a sort of S_C. Two cases can be distinguished:

1. $s_C \notin f_S(S_0)$. $X_{C_{s_C}} = X_{1_{s_C}}$ follows directly by construction. Since (2) is a pushout, $X_{3_{s_3}} \cong X_{1_{s_C}}$ with $c_S'(s_3) = s_C$ holds.

2. $s_C \in f_S(S_0)$. Pushout (2) implies $s_C \in f_S(b_S'(S_2)) = c_S'(f_S''(S_2))$ and pushout (1) implies $s_C \in f_S(b_S(S_B)) = c_S(f_S'(S_B))$. By construction, for every variable $x_C \in X_{C_{s_C}} \subseteq X_{1_{s_C}}$ two cases can be distinguished, where $s_0 \in S_0 : f_S(s_0) = s_C$:

(a) $x_C \in X_{1_{s_C}} \backslash f_{X_{s_0}}(X_{0_{s_0}})$. Analogous to the proof for the sorts above (*).

(b) $x_C \in f_{X_{s_0}}(b_{X_{s_0}}(X_{B_{s_0}}))$. Analogous to the proof for the sorts above (**).

c^*_A obviously exists (and is unique) since algebra homomorphisms are restricted to isomorphisms.
The morphism c^* is compatible with *pre*, *post*, *type* and *cond* functions since $c^* = c'^{-1} \circ c$ and c is well-defined and c'^{-1} is well-defined for $c(AN_C)$.

(3) is a pushout. First of all, the commutativity of (3) is shown. Recapitulating, $b' \circ b^* = b$, $c' \circ c^* = c$, $c' \circ f'' = f \circ b'$ and $f \circ b = c \circ f'$ hold.

$$\begin{aligned} c' \circ f'' \circ b^* &= f \circ b' \circ b^* \\ &= f \circ b \\ &= c \circ f' \\ &= c' \circ c^* \circ f' \end{aligned}$$

The fact that $c' \in \mathcal{M}$ (is a monomorphism in **AHLNet**) implies the commutativity of (3). The facts that $S_3 = S_2 +_{S_B} S_C$ and $OP_3 = OP_2 +_{OP_B} OP_C$ are pushouts in **Sets** and $E_3 = f''^{\#}_{SP}(E_2) \cup c^{*\#}_{SP}(E_C)$ remain to be shown.
(1+2) and (2) are pullbacks for the mentioned sets (sorts and operations) since pushouts along $\mathcal{M}$-morphisms in (weak) adhesive HLR categories are pullbacks (see Theorem 4.26 (1.) in [EEPT06]) and **Sets** is an adhesive category (see Definition 4.5 and Theorem 4.6 in [EEPT06]). The $\mathcal{M}$ Pushout-Pullback Decomposition Lemma (see [EEPT06], Theorem 4.26) implies that (1) is a pushout for the mentioned sets, i.e. (1) is a pushout in **Sig**.
This is analogous for the additional variables X.
To show that $E_3 = f''^{\#}_{SP}(E_2) \cup c^{*\#}_{SP}(E_C)$, suppose $\exists e_3 \in E_3 \backslash (f''^{\#}_{SP}(E_2) \cup c^{*\#}_{SP}(E_C))$. Therefore, $\exists e_1 = c'^{\#}_{SP}(e_3) \in E_1$. Let $sorts(e)$ be the set of all sorts of equation e and $opns(e)$ be the set of all operations of the equation e. Since (1) is a pushout, two cases can be distinguished:

1. $e_1 \in f^{\#}_{SP}(E_0)$. Obviously, the facts that $sorts(e_1) \subseteq f_S(S_0) \cap c'_S(S_3)$ and $opns(e_1) \subseteq f_{OP}(OP_0) \cap c'_{OP}(OP_3)$ hold. Pushout (2) implies that $sorts(e_1) \subseteq f_S(b'_S(S_2)) \cap c'_S(f''_S(S_2))$ and analogous for operations. The strictness of b' implies that $\exists e_2 \in E_2 : f^{\#}_{SP}(b'^{\#}_{SP}(e_2)) = e_1$. The well-definedness of f'' and the commutativity of (2) lead to $\exists e'_3 \in E_3 : f''^{\#}_{SP}(e_2) = e'_3 \wedge c'^{\#}_{SP}(e'_3) = e_1$. Injectivity of w and $c'^{\#}_{SP}(e_3) = e_1 = c'^{\#}_{SP}(e'_3)$ imply that $e'_3 = e_3$, which is a contradiction to the assumption that $e_3 \in E_3 \backslash (f''^{\#}_{SP}(E_2) \cup c^{*\#}_{SP}(E_C))$.

2. $e_1 \in c^{\#}_{SP}(E_C) = c'^{\#}_{SP}(c^{*\#}_{SP}(E_C))$. Then $\exists e_C \in E_C, e'_3 \in E_3 : c^{*\#}_{SP}(e_C) = e'_3 \wedge c'^{\#}_{SP}(e'_3) = e_1$. The injectivity of w and $c'^{\#}_{SP}(e_3) = e_1 = c'^{\#}_{SP}(e'_3)$ lead to $e'_3 = e_3$. This is a contradiction to the assumption that $e_3 \in E_3 \backslash (f''^{\#}_{SP}(E_2) \cup c^{*\#}_{SP}(E_C))$.

Trivially, $A_3 = A_2 +_{A_B} A_C$ is the amalgamated sum (see Lemma B.17) of A_2 and A_C with respect to A_B. As shown in Fact 3.24, (3) is a pushout for the net part. So (3) is a pushout in **AHLNet**. □

Fact 4.29 ($\mathcal{M}'$ in $(\mathbf{AHLNet}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is closed under POs and PBs along $\mathcal{M}$-Morphisms)**.** See Definition 2.11.

$$\begin{array}{ccc} AN_1 & \xrightarrow{f} & AN_2 \\ \downarrow{\scriptstyle g \in \mathcal{M}} & (1) & \downarrow{\scriptstyle g' \in \mathcal{M}} \\ AN_3 & \xrightarrow{f'} & AN_4 \end{array}$$

Proof. **Pushout.** Let (1) be a pushout and $f \in \mathcal{M}'$. Since this property holds for the net part (see Fact 3.26), it is sufficient to show that $f'_S \in \mathcal{M}'$ and $f'_{OP} \in \mathcal{M}'$. Since pushouts in **Spec** are constructed componentwise (for sorts and operations) in **Sets**, which is an adhesive category (see Definition 4.5 and Theorem 4.6 in [EEPT06]), $f'_S \in \mathcal{M}'$ and $f'_{OP} \in \mathcal{M}'$ hold. Remember the mapping f_X of the additional variables is restricted to injective morphisms.
Pullback. This follows from standard category theory (pullbacks preserve monomorphisms). □

Fact 4.30 ($\mathcal{Q}$ in $(\mathbf{AHLNet}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is closed under POs and PBs along $\mathcal{M}$-Morphisms)**.** See Definition 2.12. Analogous to Fact 4.29 since $\mathcal{Q}$=$\mathcal{M}'$.

Fact 4.31 ($(\mathbf{AHLNet}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Induced Pullback-Pushout Property for $\mathcal{M}$ and $\mathcal{Q}$)**.** See Definition 2.13.

Proof. Given the following pullback and pushout with $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ and $SP_i = (S_i, OP_i, E_i)$ for $i = 0..4$, $h \in \mathcal{M}$ and $h' \in \mathcal{Q}$ in $(\mathbf{AHLNet}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ then the induced morphism $x : AN_3 \to AN_4$ with $x \circ g' = h$ and $x \circ f' = h'$ is a monomorphism in $\mathcal{Q}$.

$$\begin{array}{ccc} AN_0 & \xrightarrow{f} & AN_1 \\ \downarrow{\scriptstyle g} & (PB) & \downarrow{\scriptstyle h} \\ AN_2 & \xrightarrow{h'} & AN_4 \end{array} \qquad \begin{array}{ccc} AN_0 & \xrightarrow{f} & AN_1 \\ \downarrow{\scriptstyle g} & (PO) & \downarrow{\scriptstyle g'} \\ AN_2 & \xrightarrow{f'} & AN_3 \end{array}$$

Since this fact holds for the net part (see Fact 3.28), the injectivity of $x_{SP} = (x_S, x_{OP}, x_X)$ remains to be shown. The proof for (x_S, x_{OP}) is analogous to the proof of the mentioned fact for N being S or OP. Remember that x_X is always injective. x is a well-defined **AHLNet**-morphism since x is induced by pushout (PO). □

Fact 4.32 ($(\mathbf{AHLNet}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Composition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$)**.** See Definition 2.14.

Proof. This follows from standard category theory (composition property of monomorphisms). □

Fact 4.33 ((**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Decomposition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$)**.** See Definition 2.15. This follows from standard category theory (decomposition property of monomorphisms).

Fact 4.34 ($\mathcal{Q}$ in (**AHLNet**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) is closed under Composition and Decomposition)**.** See Definition 2.16. Analogous to Fact 4.32 and Fact 4.33 since $\mathcal{Q}$=$\mathcal{M}'$.

4.3 AHL Systems as weak Adhesive HLR Category with NACs

In this section is shown, that the category **AHLSystems** is a weak adhesive HLR category with NACs.

Theorem 4.35 ((**AHLSystems**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) is a Weak Adhesive HLR Category with NACs)**.** *The category* (**AHLSystems**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) *with the following morphism classes is a weak adhesive HLR category with NACs:*

$\mathcal{M}$	:	*strict AHL morphisms (see Definition 4.16)*
$\mathcal{M}'$	:	*injective AHL morphisms (see Definition 4.14 and Fact B.27)*
$\mathcal{Q}$	:	*injective AHL morphisms (see Definition 4.14 and Fact B.27)*
$\mathcal{E}'$	:	*jointly equation strict and minimal jointly surjective AHL morphisms (see Definitions 4.7 and 4.20)*

Proof. As already mentioned, (**AHLSystems**,$\mathcal{M}$) is a weak adhesive HLR category (see Fact 4.18). Therefore, merely the additional properties of a *weak adhesive HLR category with NACs* (see Definition 2.4) need to be proven. These properties are proven in the following Facts 4.36, 4.38, 4.40, 4.41, 4.42 and 4.43. □

Fact 4.36 ((**AHLSystems**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Unique Epi-$\mathcal{M}$ Factorization)**.** See Definition 2.5. The Construction is analogous to Fact 4.23 with the additional marking M_3 of AN_3 with $M_3 := M_2$ for M_2 being the marking of AN_2.

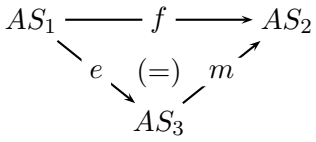
Proof. Trivial, since this fact is already proven for **AHLNet** (see Fact 4.23). □

Analogous to the case of **AHLNet**, a unique $\mathcal{E}_0$-$\mathcal{M}'$ factorization (corresponding to the $\mathcal{E}_0$-$\mathcal{M}_0$ factorization in Remark 5.26 in [EEPT06]) is defined in the next step. This and the fact that **AHLSystems** has binary coproducts (see Fact B.28) are used in the proof that **AHLSystems** has unique $\mathcal{E}'$-$\mathcal{M}'$ pair factorization.

Fact 4.37 ((**AHLSystems**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Unique $\mathcal{E}_0$-$\mathcal{M}'$ Factorization)**.** Let $\mathcal{E}_0$ be the class of equation strict (see Definition 4.6) and minimal surjective (see Definition 4.19) AHL morphisms.

Construction. The construction is analogous to the construction in **AHLNet** (see Fact 4.24). The marking M_3 of the constructed AHL system $AS_3 = (AN_3, M_3)$ is uniquely determined by the minimal property of morphism e:

$$M_3(a_3, p_3) = max(\{M_1(a,p) | a \in e_A^{-1}(a_3) \wedge p \in e_P^{-1}(p_3)\})$$

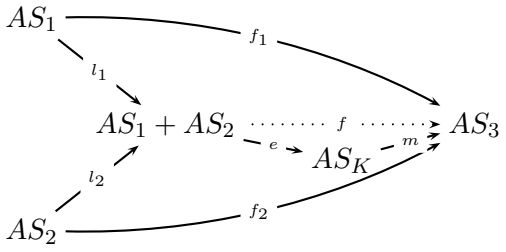


Proof. Fact 4.24 shows this property in **AHLNet**. Marking preservation of e and m and $e \in \mathcal{E}_0$ and $m \in \mathcal{M}'$ follow directly by construction. The unique isomorphism $i = (i_{SP}, i_P, i_T, i_A) : AN_3 \to AN_3'$ with $i_{SP} = (i_S, i_{OP}, i_X)$ and $i \circ e = e'$ and $m' \circ i = m$ is induced in **AHLNet** as already shown in the proof of Fact 4.24. $M_3 = M_3'$ follows directly from the the minimal property of e and e'. □

Fact 4.38 ((**AHLSystems**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Unique $\mathcal{E}'$-$\mathcal{M}'$ Pair Factorization)**.** See Definition 2.6.

Construction. The construction is analogous to the construction in **AHLNet** (see Fact 4.25). The marking M_K of the constructed AHL system $AS_K = (AN_K, M_K)$ is uniquely determined by $\mathcal{E}_0$-$\mathcal{M}'$ factorization:

$$\begin{aligned} M_K(a_K, p_K) \quad = max(\quad & \{M_1(a,p) | a \in l_{1_A}^{-1}(e_A^{-1}(a_3)) \wedge p \in l_{1_P}^{-1}(e_P^{-1}(p_3))\} \\ \cup \quad & \{M_2(a,p) | a \in l_{2_A}^{-1}(e_A^{-1}(a_3)) \wedge p \in l_{2_P}^{-1}(e_P^{-1}(p_3))\}) \end{aligned}$$



Proof. Fact 4.25 shows this property in **AHLNet**. $(e_1 = e \circ l_1, e_2 = e \circ l_2) \in \mathcal{E}'$ (are minimal) follows from the fact that $e \in \mathcal{E}_0$ (is minimal) and l_1 and l_2 are marking strict.
For the universal property, the fact that $[e_1', e_2'] \in \mathcal{E}_0$ (is minimal) remains to be shown. This is trivial since coproduct inclusions are marking strict. Therefore, $(AN_K', [e_1', e_2'], m')$ is the $\mathcal{E}_0$-$\mathcal{M}'$ factorization of f in **AHLSystems**, which is unique up to isomorphism. □

Analogous to the case without data types, there is a connection between pushouts in **AHLNet** and in **AHLSystems**. The next fact formalizes this connection. It is required for some of the following proofs.

Fact 4.39 (**AHLNet-AHLSystems** Pushout Equivalence)**.** Given the following commutative diagram (1) with $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 0..3$ in $(\mathbf{AHLSystems}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ with $m, n \in \mathcal{M}$, then the following statements hold:

1. (1) is a pushout in **AHLSystems** if (1) is a pushout in **AHLNet** and all places in P_1 without a preimage in P_0 are mapped strictly by g_P, i.e.

$$\forall (a_1, p_1) \in (A_1 \otimes (P_1 \backslash m_P(P_0))) : M_1(a_1, p_1) = M_3(g_A(a_1), g_P(p_1))$$

2. (1) is a pushout in **AHLNet** if (1) is a pushout in **AHLSystems**.

$$\begin{array}{ccc} AS_0 & \xrightarrow{m} & AS_1 \\ \downarrow{\scriptstyle f} & (1) & \downarrow{\scriptstyle g} \\ AS_2 & \xrightarrow{n} & AS_3 \end{array}$$

Proof. The proof of this fact is analogous to the proof of the corresponding fact for the case without data types (see Fact 3.20). □

Fact 4.40 ($(\mathbf{AHLSystems}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has $\mathcal{M}$-$\mathcal{M}'$ Pushout-Pullback Decomposition Property)**.** See Definition 2.7.

Proof. Since this fact is already shown for AHL nets (see Fact 4.26) and pullbacks (1) and (2) follow from the categorical properties of (weak) adhesive HLR categories (see proof of Fact 4.26), the fact that (1) is a pushout in **AHLSystems** remains to be proven. By Fact 4.39, it is sufficient to show $\forall (a_3, p_3) \in A_3 \otimes (P_3 \backslash l_P(P_0)) : M_3(a_3, p_3) = M_4(u_A(a_3), u_P(p_3))$. The following diagram shows the expansion of the construction in AHL Nets:

$$\begin{array}{ccccc} (AN_0, M_0) & \xrightarrow{k} & (AN_1, M_1) & \xrightarrow{r} & (AN_2, M_2) \\ \downarrow{\scriptstyle l} & (1) & \downarrow{\scriptstyle z} & (2) & \downarrow{\scriptstyle v} \\ (AN_3, M_3) & \xrightarrow{u} & (AN_4, M_4) & \xrightarrow{w} & (AN_5, M_5) \end{array}$$

Let $(a_3, p_3) \in A_3 \otimes (P_3 \backslash l_P(P_0))$. Since (1+2) is a pushout in **AHLSystems**, $M_3(a_3, p_3) = M_5(w_A(u_A(a_3)), w_P(u_P(p_3)))$ hold. Additionally, it is valid that

1. $M_3(a_3, p_3) \leq M_4(u_A(a_3), u_P(p_3))$ and
2. $M_4(u_A(a_3), u_P(p_3)) \leq M_5(w_A(u_A(a_3)), w_P(u_P(p_3)))$.

$$M_3(a_3,p_3) \leq M_4(u_A(a_3),u_P(p_3)) \leq M_5(w_A(u_A(a_3)),w_P(u_P(p_3))) = M_3(a_3,p_3)$$
$$\Rightarrow M_3(a_3,p_3) = M_4(u_A(a_3),u_P(p_3))$$

Pushout decomposition implies that (2) is a pushout. □

Fact 4.41 ((**AHLSystems**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has $\mathcal{M}$-$\mathcal{Q}$ Pushout-Pullback Decomposition Property)**.** See Definition 2.8. Analogous to Fact 4.40 since $\mathcal{Q}$=$\mathcal{M}'$.

Fact 4.42 ((**AHLSystems**, $\mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q}$) has Initial Pushouts over $\mathcal{M}'$-Morphisms)**.** See Definition 2.10.

Construction. Given AHL systems $AS_i = (AN_i, M_i)$ with $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 0, 1$ and an injective **AHLSystems**-morphism $f : AS_0 \to AS_1 \in \mathcal{M}'$, then the boundary AS_B can be constructed analogously to the boundary net in **AHLNet** (see Fact 4.28) except for the set P_B:

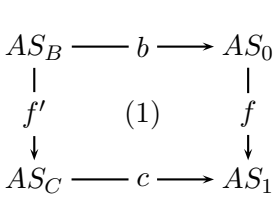
- $P_B = P_{B_{\mathbf{AHLNet}}} \cup \{p \in P_0 | \exists a \in A_0 : M_0(a,p) < M_1(f_A(a), f_P(p))\}$, where $P_{B_{\mathbf{AHLNet}}}$ is the set P_B of the boundary in **AHLNet** (see Fact 4.28).
- $\forall (a,p) \in (A_B \otimes P_B) \subseteq (A_0 \otimes P_0) : M_B(a,p) = M_0(a,p)$.

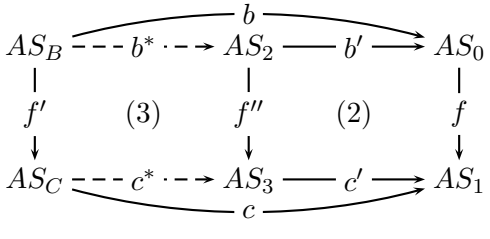
The boundary net in **AHLSystems** differs from the boundary net in **AHLNet** because it additionally contains all places that are not mapped strictly by f. This construction is analogous to the case without data types (see Fact 3.25).

The context object $AS_C = (AN_C, M_C)$ with $AN_C = (SP_C, P_C, T_C, pre_C, post_C, cond_C, type_C, A_C)$ with $SP_C = (S_C, OP_C, E_C, X_C)$ can be constructed as pushout complement in **AHLSystems**, analogous to the pushout complement in **AHLNet** with:

$$\forall (a,p) \in (A_C \otimes P_C) \subseteq (A_1 \otimes P_1) : M_C(a,p) = M_1(a,p)$$

The construction of the strict morphisms $b : PS_B \to PS_0 \in \mathcal{M}$, $c : PS_C \to PS_1 \in \mathcal{M}$ and the morphism $f' : PS_B \to PS_C \in \mathcal{M}'$ is analogous to the construction in **AHLNet**. This construction leads to *initial pushout* (1).





Proof. This is analogous to the case without data types since this is already proven for **AHLNet** (see Fact 4.28) and the expansion of Fact 3.20 for P/T nets to AHL nets holds (see Fact 4.39). □

Fact 4.43 (Additional Properties of $(\mathbf{AHLSystems}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$)**.** The category $(\mathbf{AHLSystems}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has the following properties:

1. $\mathcal{M}'$ is closed under POs and PBs along $\mathcal{M}$-Morphisms (see Definition 2.11).
2. $\mathcal{Q}$ is closed under POs and PBs along $\mathcal{M}$-Morphisms (see Definition 2.12).
3. Induced Pullback-Pushout Property for $\mathcal{M}$ and $\mathcal{Q}$ (see Definition 2.13).
4. Composition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$ (see Definition 2.14).
5. Decomposition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$ (see Definition 2.15).
6. $\mathcal{Q}$ is closed under Composition and Decomposition (see Definition 2.16).

Proof. This follows directly by the fact that $(\mathbf{AHLNet}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is a weak adhesive HLR category with NACs with the morphism classes presented in Theorem 4.22. □

4.4 Case Study: AHL Airport Control System

4.4.1 Overview

This chapter extends the airport control system ACS presented in Section 3.3 to AHL systems. This system is called *AHL-ACS*. Analogous to ACS, it is supposed to prevent accidents in the airport area and, additionally to ACS, it is also responsible for the coordination of the airplanes at the gates. The system can handle airplanes and gates of different sizes and manage the coordination of the airplanes at the gates. In this chapter, AHL-ACS is described in detail and modeled as reconfigurable AHL system with negative application conditions. Analogous to ACS, transforming the AHL system represents the rearrangement of the airport and firing of a transition reflects a process at the airport.

The example is divided into different parts. First of all, the requirement engineering of AHL-ACS is presented. Afterwards, the specifications and algebras are described. Finally, the AHL systems and rules are introduced.

4.4.2 Detailed Description

In principle, AHL-ACS has the same functionality as ACS with some additional features. To simplify matters, all functionalities also modeled in ACS are neglected in this example. So the airport is restricted to a level-2-airport with one starting runway, one landing runway and space for up to five airplanes at the gate area. The neglected functionalities could be modeled analogous to the case without data types (see Section 3.3).
The capacity of the gate area corresponds to the number of airplanes being able to use the gate area simultaneously. In the case of ACS, this number is called number

of gates.

In AHL-ACS, the representation of an airplane consists of an ID and a fixed size. AHL-ACS distinguishes seperate gates and ensures that a gate can only be used exclusively by one airplane. Every gate has a fixed size. In the case of AHL-ACS, size of a gate means that only airplanes of exactly this size are allowed to use this certain gate. Only one gate of size 1 is available in the basic AHL-ACS (i.e. in the initial situation).
AHL-ACS ensures that only airplanes of supported sizes are allowed to use the airport. A size is called supported if at least one gate of this size exists. Airplanes with unsupported sizes are neither allowed to enter the airport nor the airspace of the airport.
The possibility to change the number of gates while running is also provided by AHL-ACS. Additionally, new sizes of airplanes can be created and used in the system. A detailed description of these expansions follows.

Adding new sizes of airplanes is always possible, provided that the new size does not already exist. Added (and not later removed) sizes and the size available in the basic AHL-ACS are called *known sizes*. Removing sizes of airplanes is only allowed if the airport does not support this size. These operations correspond to an updating of the airplane data base and does not affect the airport directly at this moment.
Furthermore, adding new gates of known sizes is always possible. However, the following condition has to be satisfied: If the new gate is the first gate of this size, this size has to be added to the supported sizes of the airport. Neither the number of gates nor the number of sizes is limited by AHL-ACS.
In contrast to this expansion of the airport, it is also possible to remove gates. Therefore, the following constraints have to hold:

- A gate can only be removed if it is unused.
- A gate of a size can be removed if another gate of this size exists.
- The last gate of a size can only be removed if at least one other gate (of any size) exists and no airplane of this size is at the airport (including the airspace of the airport). In this case, the support of this size is removed as well.

4.4.3 Systems and Rules

AHL-ACS is given by the reconfigurable AHL system with NACs $AHL\text{-}ACS = (Lvl2ap, AHL\text{-}ACS\text{-}RULES)$ with the AHL system $Lvl2ap$ and the set of rules $AHL\text{-}ACS\text{-}RULES$ presented in this subsection.
First of all, the required specifications and algebras are defined.

Specifications and Algebras

SP-ACS0 =

sorts:	$airplane$,
	$apSize$,
	$blackToken$
opns:	$\bullet :\rightarrow blackToken$,
	$getSize : airplane \rightarrow apSize$
vars:	$b_1 : blackToken$
eqns:	$b_1 = \bullet$

$$\begin{aligned} A_{SP\text{-}ACS0} &= ((\mathbb{N}^+ \times \mathbb{N}^+), \mathbb{N}^+, \{\bullet\}, \bullet, (a, s) \overset{getSize}{\longmapsto} s) \\ A'_{SP\text{-}ACS0} &= ((\mathbb{N}^+ \times \mathbb{S}), \mathbb{S}, \{\bullet\}, \bullet, (a, s) \overset{getSize}{\longmapsto} s) \text{ with} \\ \mathbb{S} &= \{s_i | i \in \mathbb{N}^+\} \end{aligned}$$

Specification **SP-ACS0** forms the base of AHL-ACS. The sort $apSize$ stands for *airplane size*. Remaining sorts $blackToken$ and $airplane$ and the operation $getSize$ are self-explanatory. The equation $b_1 = \bullet$ ensures that the carrier set of the sort $blackToken$ of every **SP-ACS0**-algebra is a singleton, i.e. a set with exactly one element.
$A_{SP\text{-}ACS0}$ and $A'_{SP\text{-}ACS0}$ are algebras to this specification, where $\mathbb{N}^+ = \mathbb{N}\backslash\{0\}$ is the set of all positive integers. Algebra $A_{SP\text{-}ACS0}$, used in the AHL system, and algebra $A'_{SP\text{-}ACS0}$, used in some rules, differ in order to express that various mappings are possible. These and the following specifications and algebras marked with an apostrophe (') are only used in the transformation rules. Remember that algebra homomorphisms are restricted to isomorphisms.

SP-ACS1' = SP-ACS0 +

sorts:	$\emptyset$
opns:	$size :\rightarrow apSize$
vars:	$\emptyset$
eqns:	$\emptyset$

$$A_{SP\text{-}ACS1'} = A'_{SP\text{-}ACS0} \text{ with the additional constant } (s_1)$$

SP-ACS2' = SP-ACS1' +

sorts:	$\emptyset$
opns:	$size' :\rightarrow apSize$
vars:	$\emptyset$
eqns:	$\emptyset$

$$A_{SP\text{-}ACS2'} = A_{SP\text{-}ACS1'} \text{ with the additional constant } (s_2)$$

SP-ACS2" = SP-ACS2' +
sorts: $\emptyset$
opns: $size'' :\rightarrow apSize$
vars: $\emptyset$
eqns: $\emptyset$

$A_{SP\text{-}ACS2''}$ = $A_{SP\text{-}ACS2'}$ with the additional constant (s_2)

SP-ACS1 = SP-ACS0 +
sorts: $\emptyset$
opns: $size_1 :\rightarrow apSize$
vars: $\emptyset$
eqns: $\emptyset$

$A_{SP\text{-}ACS1}$ = $A_{SP\text{-}ACS0}$ with the additional constant (1)

SP-ACS2 = SP-ACS1 +
sorts: $\emptyset$
opns: $size_2 :\rightarrow apSize$
vars: $\emptyset$
eqns: $\emptyset$

$A_{SP\text{-}ACS2}$ = $A_{SP\text{-}ACS1}$ with the additional constant (2)

SP-ACS = SP-ACS1 +
sorts: nat
opns: $getID : airplane \rightarrow nat$
vars: $\emptyset$
eqns: $\emptyset$

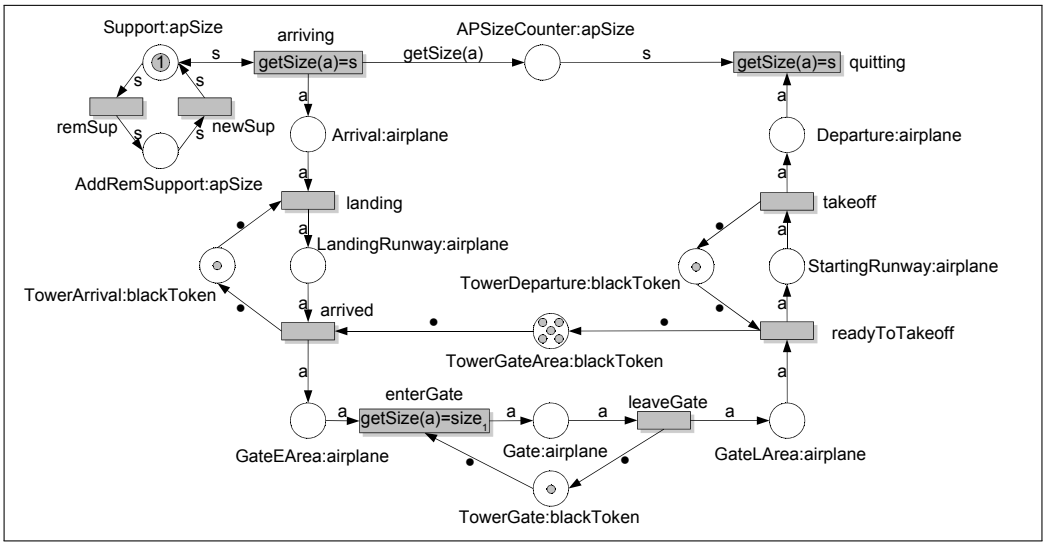
$A_{SP\text{-}ACS}$ = $A_{SP\text{-}ACS1}$ with the additional carrier set and operation $(\mathbb{N}^+, (a, s) \overset{getID}{\longmapsto} a)$

SP-ACSa = SP-ACS +
sorts: $\emptyset$
opns: $size_2 :\rightarrow apSize$
vars: $\emptyset$
eqns: $\emptyset$

$A_{SP\text{-}ACSa}$ = $A_{SP\text{-}ACS}$ with the additional constant (2)

Specification **SP-ACS** and algebra $A_{SP\text{-}ACS}$ are used in the start system $Lvl2ap$. The additional sort nat and the additional operation $getID$, assigning a unique identification to every airplane, are defined to distinguish between different airplanes (of the same size) in the AHL system. Neither this sort nor this operation are used in the transformation rules.
Note that some of the presented specifications and algebras are only required for showing theoretical results to AHL-ACS in Subsection 4.4.4.

Startsystem: *Lvl2ap*



This picture shows the start system of AHL-ACS with specification **SP-ACS** and algebra $A_{SP\text{-}ACS}$. Note that $a \in X_{airplane}$ and $s \in X_{apSize}$ in this and all following presented AHL-ACS nets. The place *Support* contains exactly one token of each supported size. Only size 1 is supported in the initial situation. The transitions *newSup* and *remSup* should only be fired before resp. after changing the supported sizes and the place *AddRemSupport* should not be used in the regular operation of AHL-ACS. It is only required for adding and removing supported sizes.
Firing transition *arriving* represents the arrival of an airplane in the airspace of the airport (place *Arrival*). In contrast to ACS, this transition has a nonempty predomain and a firing condition. It is only enabled if the size of the arriving airplane a equals size s and there is a token of this size at the place *Support*. This ensures that only airplanes of supported sizes are allowed to enter the airspace and use the airport. For each arrived airplane, an airplane token is placed at *Arrival* and a size token of the size of the airplane is placed at $APSizeCounter$. This place represents a counter for all airplanes at the airport and stores their sizes. If an airplane leaves the airport a token of the corresponding size is removed from this place. This is represented by firing transition *quitting*.
The starting and the landing runways are modeled as in ACS. Note that the number of runways cannot be changed in this example. The place $TowerGateArea$ ensures that only five airplanes can use the gate area simultaneously. In this example, the gate area represents the size of the whole gate area, no matter how many gates exist. To simplify matters, the size of the gate area is fixed. Note again that changing the size of the gate area could be modeled analogously to the case without data types (see Rules $increaseNumberOfGates$ and $decreaseNumberOfGates$ in ACS).
The first and only gate of the airport is represented by the places *Gate* and the complement place $TowerGate$ of this gate. So the exclusive use of the gate by only one airplane is guaranteed. The condition of the transition *enterGate* ensures that

only airplanes of size 1 can use this gate.
Note that the specification **SP-ACS** has only one constant $size_1$ of the sort $apSize$. The values of the constants of sort $apSize$ form the set of the *known sizes* mentioned above.

Through different matches it is possible to create and remove arbitrary (but unused) constants and gates of arbitrary (but known) sizes without using an infinite set of rules. This is feasible although algebra homomorphisms are restricted to isomorphisms because the carrier sets are not changed.
Note that all morphisms used in the following rules are inclusions.

Rule 1: $addNewSize$

$$\begin{array}{c}\textbf{SP-ACS1'}\\ A_{SP\text{-}ACS1'}\end{array} \hookleftarrow \begin{array}{c}\textbf{SP-ACS1'}\\ A_{SP\text{-}ACS1'}\end{array} \hookrightarrow \begin{array}{c}\textbf{SP-ACS2'}\\ A_{SP\text{-}ACS2'}\end{array}$$

Adding a new size to the set of known sizes is expressed by this rule. A new constant of the sort $apSize$ is added to the airport. Its value is determined by the match. A negative application condition is required to prevent the creation of multiple constants with the same value. The second constant of the sort $apSize$ is required to prevent a mapping from sort $apSize$ to sort nat and operation $getSize$ to $getID$. Note that the applicability of this rule is ensured since it is guaranteed that at least one constant of the sort $apSize$ exists. This is because the gluing condition (see Fact B.32) restricts the removal of the last constant since the last gate cannot be removed (and the constant is used in the fire condition of its entering transition).

Rule 1 - NAC 1:

SP-ACS2'
$A_{SP\text{-}ACS2'}$

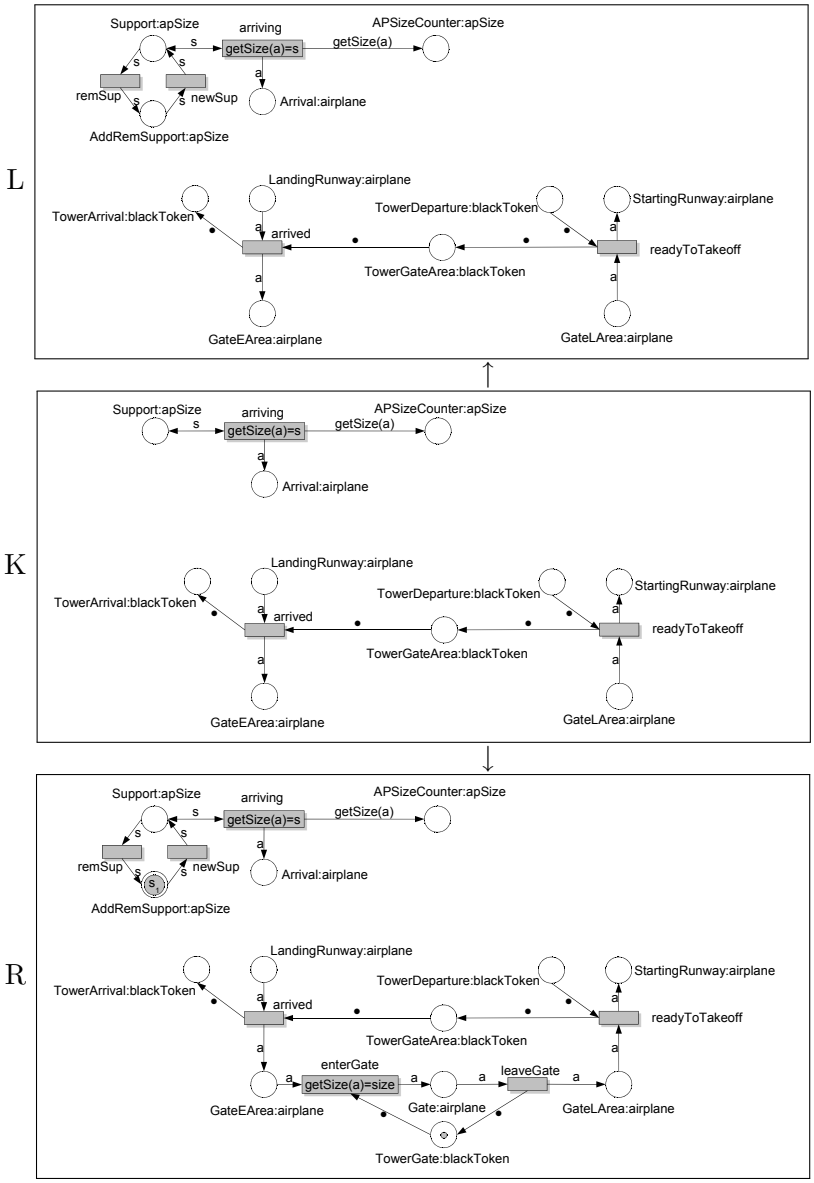
This negative application condition prevents this rule from being applied if there already exists a constant with the same value.

Rule 2: $removeSize$

Inverse to Rule 1 without negative application conditions.

The application of this rule removes a known size. This means that a constant is removed from the airport net by applying this rule. No negative application condition is required to ensure the condition that no airplane of this size is at the airport since this condition has to be satisfied before removing the last gate of a size. Additionally, the gluing condition (see Fact B.32) guarantees that no gate of this size exists.

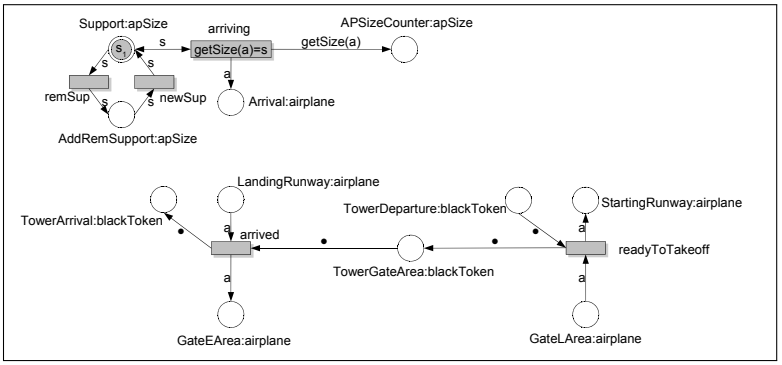
Rule 3a: *addFirstGate*



This rule expresses the adding of the first gate of an arbitrary size. As already mentioned, the size is determined by the match. The application to an airport removes the place *AddRemSupport* with the adjacent transitions and recreates these nodes with one token of the new size at the place *AddRemSupport*. Additionally, a gate of this size is added. After applying this rule, the transition *newSup* should be fired so that airplanes of the new supported size can use the airport. This complicated procedure for adding a token is necessary since $\mathcal{M}$-morphisms have to be marking strict.
Note that the gluing condition (see Fact B.32) prevents this rule from being applied if

there is a token at the place *AddRemSupport*. Nevertheless, one negative application condition is still required.

Rule 3a - NAC 1:

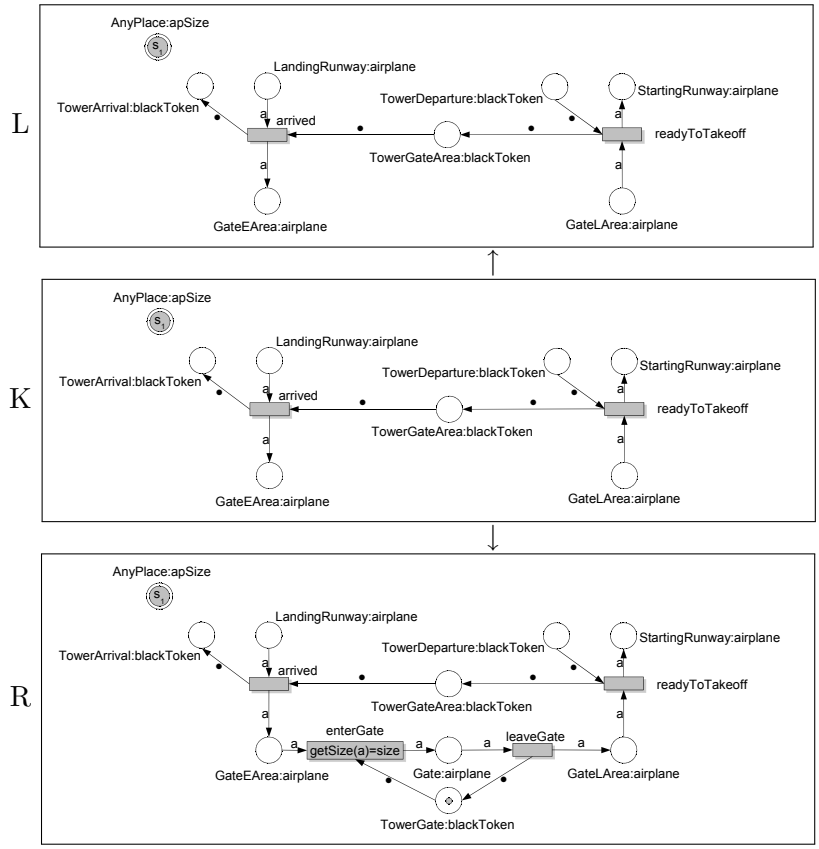


SP-ACS1'
$A_{SP\text{-}ACS1'}$

This negative application condition prevents this rule from being applied if a token of the new supported size is already at the place *Support*, i.e. the airport already supports the new size. So it is guaranteed that only the first gate of a size can be created by this rule.

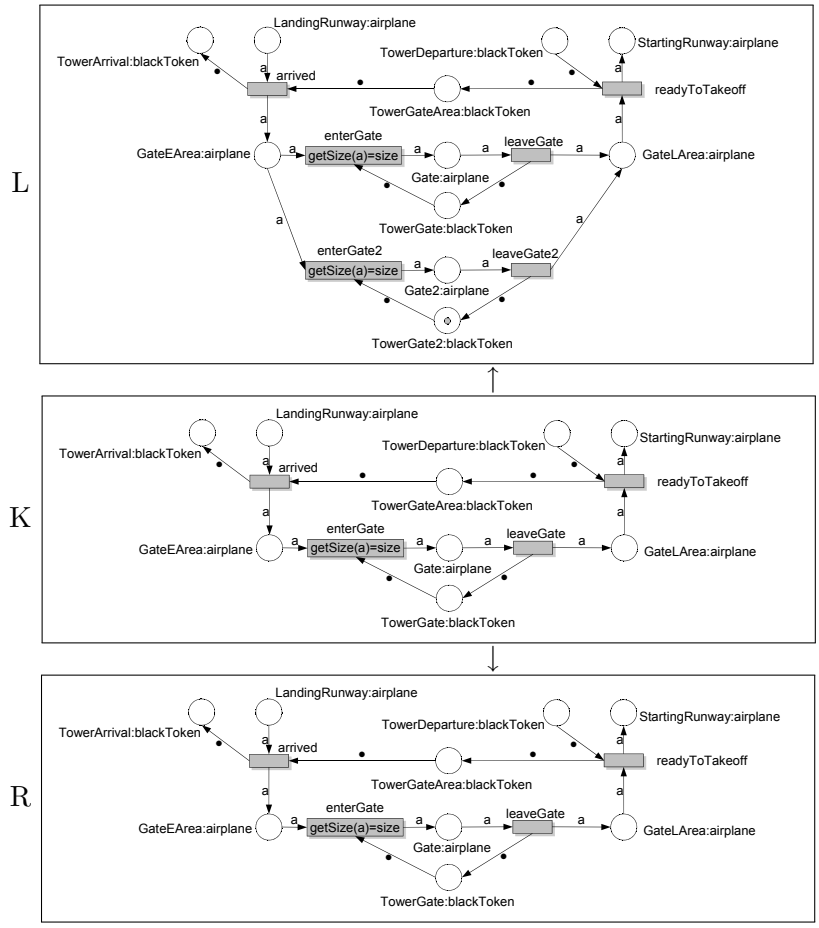
Remark 4.44. It is an important result that $\mathcal{Q}$-morphisms do not have to be marking strict. Otherwise an infinite set of NACs would be required to achieve this and the most following conditions.

Rule 3b: *addGate*



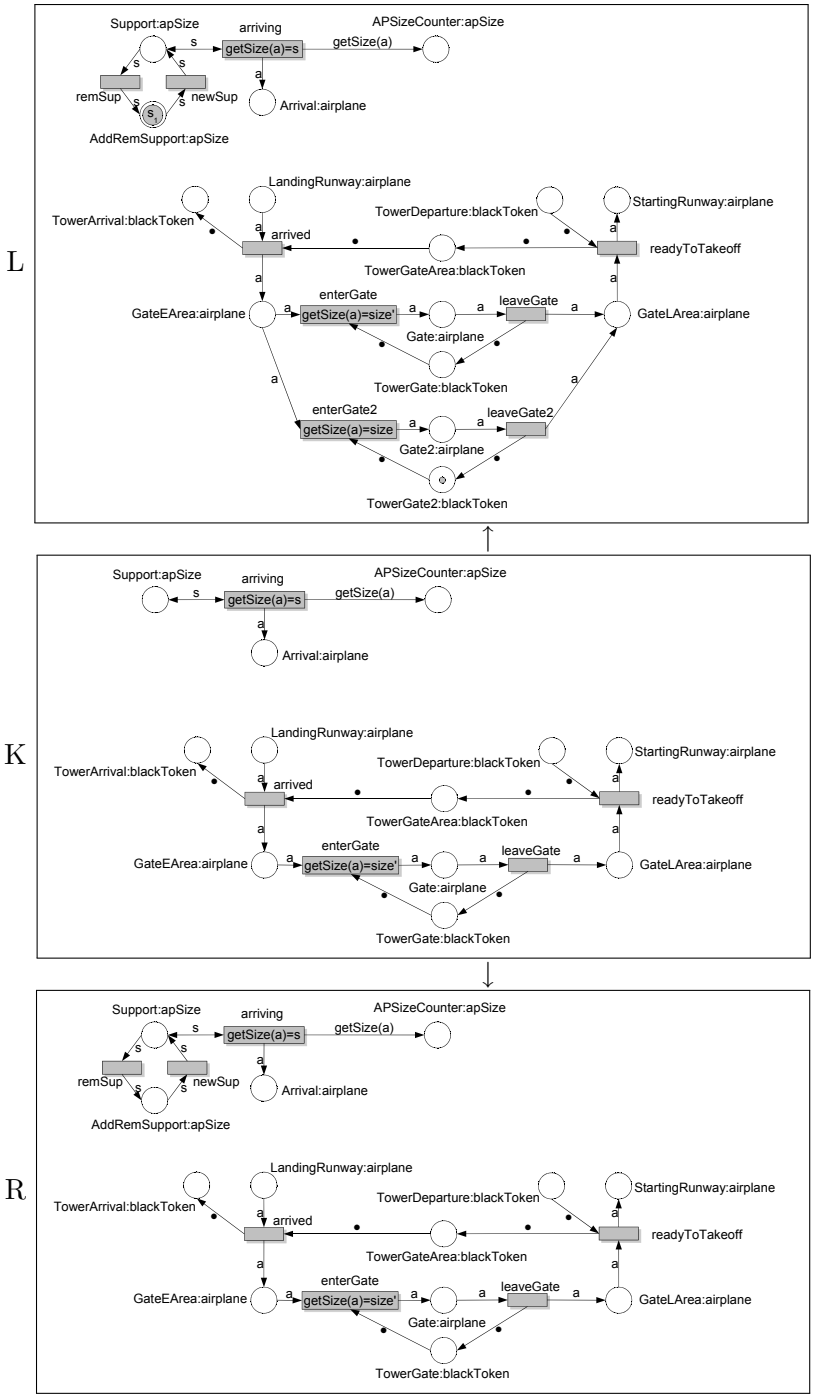
Adding a gate of an arbitrary size is modeled by this rule. However, it can only be applied if at least one gate of this size exists, i.e. the airport already supports this size. This condition is ensured by the token of the size s_1 at the place *AnyPlace*, which can be mapped arbitrarily. No negative application conditions are required for this rule.

Rule 4a: *removeGate*



This rule expresses the removal of gates, where it is not sufficient to use the inverse rule of rule 3b. The additional condition that the gate to be removed is not the last gate of a size has to be verified. Removing the last gate of a size is modeled by a different rule since this requires special treatment. The token at place $TowerGate2$ ensures that the gate to be removed is not in current use.

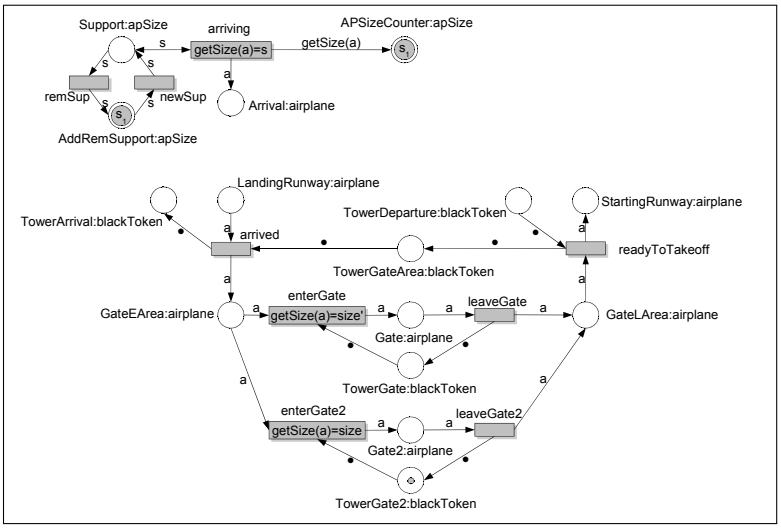
Rule 4b: *removeLastGate*



Removing the last gate of a size entails the removing of the support of this size. This is modeled by this rule. It can only be applied if there is a token of the size to be removed at the place *AddRemSupport* and at least one gate of an arbitrary but different size is left. Identifying the constants $size = s_1$ and $size' = s_2$ by a

match is not possible since this would need a non-isomorphic algebra homomorphism. Nevertheless, two negative application conditions are required.

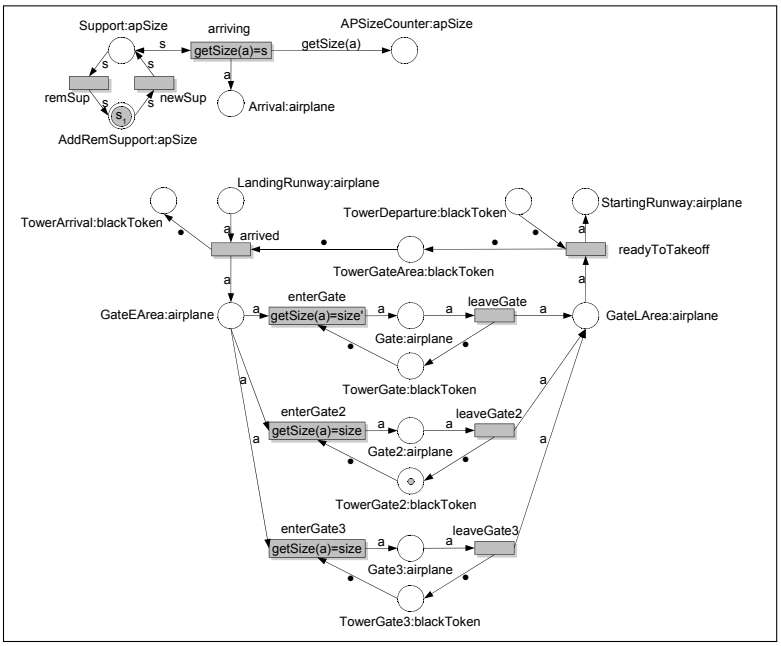
Rule 4b - NAC 1:



SP-ACS2'
$A_{SP\text{-}ACS2'}$

This negative application condition ensures that the last gate of a size can only be removed if there is no airplane of this size at the airport. This is necessary to prevent airplanes from being stuck.

Rule 4b - NAC 2:

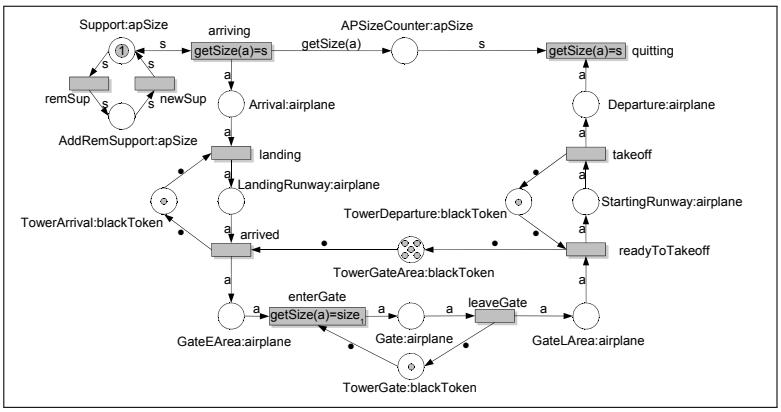


SP-ACS2'
$A_{SP\text{-}ACS2'}$

This negative application condition guarantees that this rule can only be applied if the gate to be removed is the last gate of its size.

4.4.4 Applying Theoretical Results to AHL-ACS

In the case of ACS examples for independent and dependent transformations, parallelism, concurrency and critical pairs are presented. In this section, a critical pair of a given pair of transformations is calculated and its strict NAC-confluence (see Definition 2.48) is shown. Remember that strict NAC-confluence of all critical pairs of an adhesive HLR system with NACs implies its *local confluence* (see Theorem 2.49). However, it is very complex to prove the strict NAC-confluence of critical pairs as shown in the following. Therefore, this is only demonstrated exemplarily for one critical pair. As already mentioned, a tool support in this context is necessary to perform critical pair analyses. The java-based tool AGG (see [AGG08]) provides this functionality. Note that AGG is still an ongoing project of the *TFS-group* and, therefore, still under development.
For proving the strict NAC-confluence of the following critical pair, it is necessary to construct a concurrent rule (see Definition 2.41) and a derived span (see Definition 6.9 in [EEPT06]).



SP-ACSa
$A_{SP\text{-}ACSa}$

Figure 4.1: AHL system G

Based on AHL system G, pictured in Figure 4.1, the transformations by applying rules

$$p_1 = (L_1, L_1, R_1) := removeSize$$

for removing constant $size_2$ and

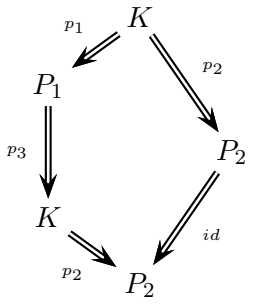
$$p_2 = (L_2, K_2, R_2) := addFirstGate$$

for adding a gate of $size_2$ are parallel dependent since there is a delete-use conflict (with respect to constant $size_2$).

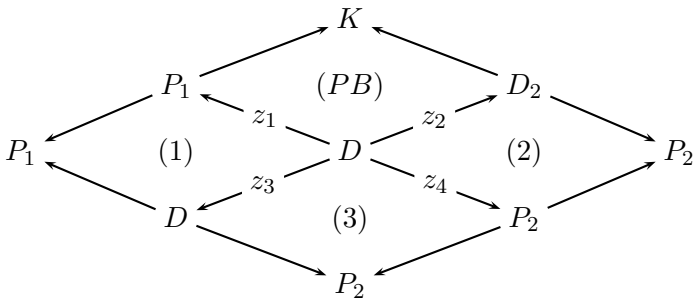
$$H_1 \overset{p_1}{\Leftarrow} G \overset{p_2}{\Rightarrow} H_2$$

The critical pair $(P_1 \overset{p_1}{\Leftarrow} K \overset{p_2}{\Rightarrow} P_2)$ of these transformations is depicted in Figure 4.2. The NACs of this critical pair are the NACs of the rules p_1 and p_2 (in this case only the NAC of p_2, called N_2).

Showing the strict NAC-confluence of a critical pair (see Definition 2.48) proceeds always the same. First, the confluence of this critical pair is shown. Note that rule *removeLastGate* cannot be applied to P_2 since no second gate exists. Nevertheless, the removed size can be added again (i.e. apply rule $p_3 := addNewSize$ to P_1) resulting in the AHL system K. Applying $p_2 = addFirstGate$ to K leads to AHL system P_2. So this critical pair is confluent.



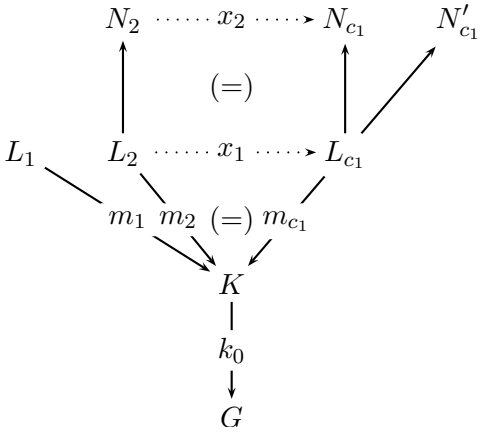
Next, the strictness of the critical pair is shown. Therefore, the derived span (see Definition 6.9 in [EEPT06]) of transformation sequence $P_1 \overset{p_3}{\Rightarrow} K \overset{p_2}{\Rightarrow} P_2$ which is given by $P_1 \leftarrow D \rightarrow P_2$ with D as pictured in Figure 4.3 needs to be calculated. Now, the pullback (PB) in the following diagram is constructed, where D_2 is the gluing object of the transformation $K \overset{p_2}{\Rightarrow} P_2$, pictured in Figure 4.4, and the pullback object D is identical to object D of the derived span shown in Figure 4.3:



The existence of morphisms $z_3 = id : D \rightarrow D$ and $z_4 : D \rightarrow P_2$ such that (2), (3) and (4) commute is obvious.

In the last step, NAC-confluence is shown. Therefore, the concurrent rule with NACs (see Definition 2.41) p_{c_1} of $K \overset{p_1}{\Rightarrow} P_1 \overset{p_3}{\Rightarrow} K \overset{p_2}{\Rightarrow} P_2$ is constructed. This rule, depicted in Figure 4.5, removes a constant of the sort *apSize* of the airport system and then adds a new constant with the same value and a new gate of this size. Its first negative application condition prevents this rule from being applied if a gate

of this size exists (i.e. there is a token of this size at place *Support*). This NAC corresponds to the down- and leftward translation (see Definition 5.6 in [Lam07]) of the NAC of rule p_2. The second negative application condition results from to the down- and leftward translation of the NAC of rule p_3 and forbids the existence of a second constant with the same value as the constant to be removed.
Let $k_0 : K \to G \in \mathcal{M}'$ be a NAC-consistent (see Definition 2.44) morphism with respect to $K \overset{p_2}{\Rightarrow} P_2$. Remember that p_1 has no negative application condition. Therefore, NAC-consistency with respect to $K \overset{p_1}{\Rightarrow} P_1$ holds for every morphism $k_0 : K \to G$.



For proving NAC-confluence, NAC-consistency of k_0 with respect to $K \overset{p_c}{\Rightarrow} P_2$ remains to be shown. This obviously holds for N_{c_1} since N_{c_1} and N_2 only differ in one additional constant. Note that $m_2 : L_2 \to K$ maps constant *size* to $size_2$ and $m_{c_1} : L_{c_1} \to K$ maps $size'$ to $size_2$. More formally, NAC-confluence follows from Theorem 6.6 in [Lam07] (where NAC N'_{c_1} is neglected) since the second concurrent rule p_{c_2} is rule p_2 itself and $x_1 : L_2 \to L_{c_1}$ and $x_2 : N_2 \to N_{c_1}$, as pictured in the diagram above, exist.
NAC-consistency of k_0 including N'_{c_1} does not hold in general since the existence of a second constant with the same value is not prohibited explicitly by the NAC of p_2. However, the creation of an AHL system G with two constants of the same value by applying rules to the start system is not possible in AHL-ACS. So NAC-confluence holds with respect to the language generated by grammar AHL-ACS (see Definition 5.4 in [EEPT06]), but not for general AHL systems. This condition is sufficient for a reconfigurable AHL system since a valid startsystem is included.

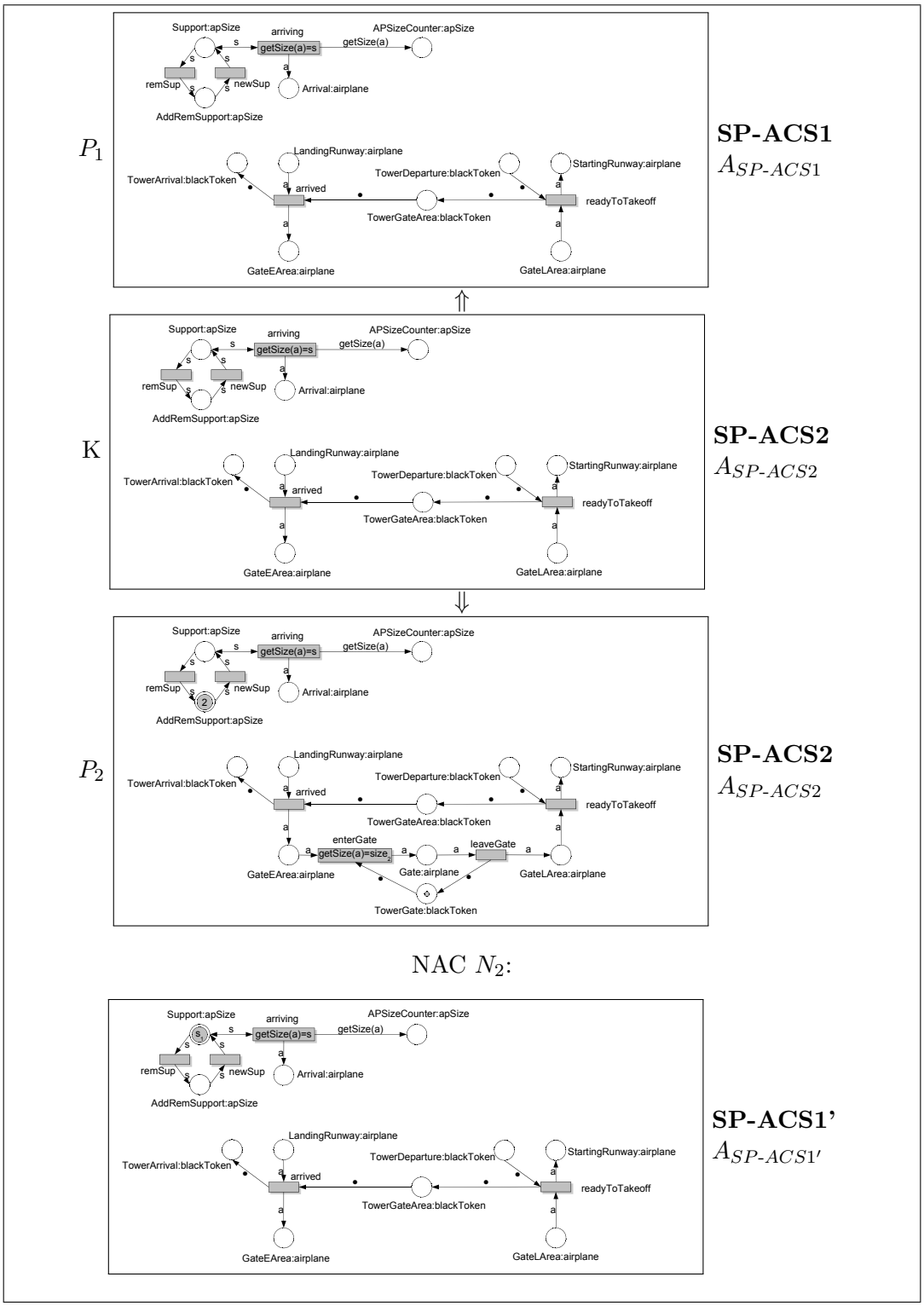
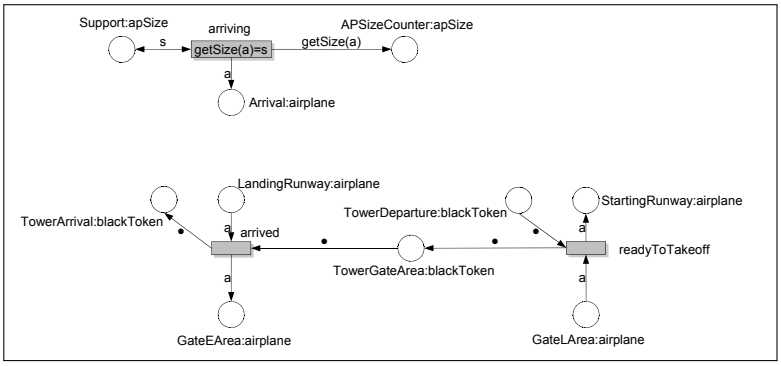
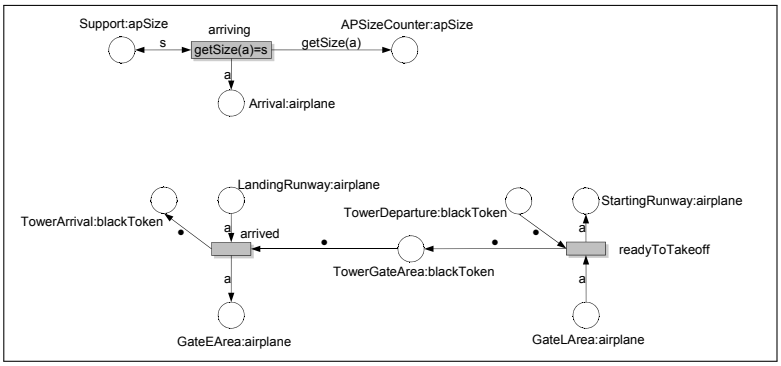


Figure 4.2: Critical Pair with NAC



SP-ACS1
$A_{SP\text{-}ACS1}$

Figure 4.3: Derived Span - Object D



SP-ACS2
$A_{SP\text{-}ACS2}$

Figure 4.4: Strictness of Critical Pair - Object D_2

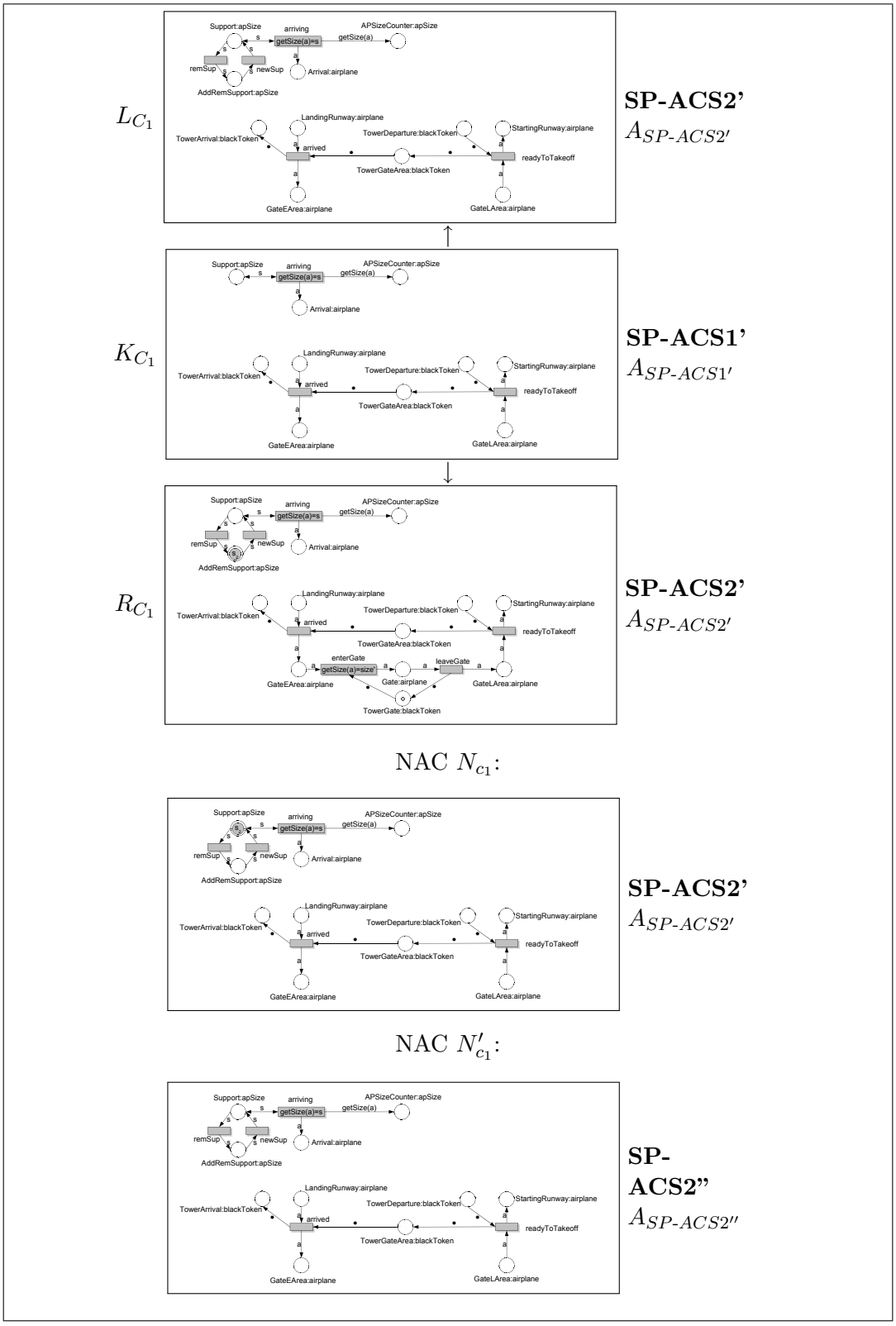


Figure 4.5: Concurrent Rule p_{c_1} with NACs

Chapter 5

Reconfigurable Labeled P/T Systems with NACs

5.1 Introduction of Reconfigurable Labeled P/T Systems

Labeled P/T systems are an extremely useful extension of P/T systems. The idea of labels is quite easy: Places obtain fixed labels and can only be mapped to places with the same label. This obviously prevents nonsensical matches. Moreover, modeling with labeled P/T systems is less error-prone than with unlabeled P/T systems since the labeled rules are usually smaller and more clear. In this thesis, only P/T systems with labeled places are considered, although, adding labels to transitions is also possible. In most cases, labeled places are sufficient for intuitive modeling and labeled transitions are not required.
Labeled P/T systems are introduced in this section and it is shown that the category **PTSys**(**L**) of L-labeled P/T systems and L-labeled P/T morphisms is a weak adhesive HLR category. Special morphisms (i.e. monomorphisms, epimorphisms, jointly epimorphic morphisms, isomorphisms) and some categorical constructions (binary coproducts, pushouts and pullbacks along $\mathcal{M}$-morphisms) of this category are defined in Appendix C.1 and the correctness of these constructions is proven.

For labeling P/T systems over an arbitrary, but *fixed set* L, a so called *labeling function* $\lambda : P \to L$, assigning a label to each place is used. The next definition introduces *L-labeled P/T systems* (short *labeled P/T systems*) formally.

Definition 5.1 (L-Labeled P/T System)**.** A L-labeled P/T system $LPS = (PS, \lambda : P \to L)$ is given by P/T system PS and a labeling function $\lambda : P \to L$ assigning a label $l \in L$ to each place $p \in P$.

L-labeled P/T morphisms (short *labeled P/T morphisms*) are P/T morphisms preserving the labeling of the places. The formal definition follows directly.

Definition 5.2 (L-Labeled P/T Morphism)**.** A L-labeled P/T morphism

$$f : (PS_1, \lambda_1) \to (PS_2, \lambda_2)$$

is a P/T morphism $f = (f_P, f_T)$ with $\lambda_1 = \lambda_2 \circ f_P$.

In the next step, the category $\mathbf{PTSys}(\mathbf{L})$ of L-labeled P/T systems and L-labeled P/T morphisms is introduced, its well-definedness is shown and it is proven that this category is a weak adhesive HLR category with the special morphism class $\mathcal{M}$ of all *strict* L-labeled P/T morphisms (see Definition 3.10).

Fact 5.3 (Category $\mathbf{PTSys}(\mathbf{L})$ is a Weak Adhesive HLR Category)**.** L-labeled P/T systems and L-labeled P/T morphisms form the category $\mathbf{PTSys}(\mathbf{L})$, where the composition of morphisms is defined as the composition of **PTSys**-morphisms and the identity is defined as identity in **PTSys**. This category is a weak adhesive HLR category (see Definition 2.3) with the special morphism class $\mathcal{M}_{strict}$ of all *strict* L-labeled P/T morphisms (see Definition 3.10).

Proof. Consider the category $\mathbf{L}$ with only one object L (an arbitrary set of labels) and only one morphism $id : L \to L$ (the identical function). $\mathbf{L}$ obviously is a well-defined category.
Construct the comma category $\mathbf{C} = ComCat(F, Incl, \{1\})$ (see Definition A.41 in [EEPT06]) with $F : \mathbf{PTSys} \to \mathbf{Sets}$, $F_{Ob}((P, T, pre, post)) = P$, $F_{Mor}((f_P, f_T)) = f_P$ and the inclusion functor $Incl : \mathbf{L} \to \mathbf{Sets}$, $Incl_{Ob}(L) = L$ and $Incl_{Mor}(id_L) = id_L$:

$$\begin{aligned} Ob_{\mathbf{C}} &= \text{class of all triples } (PS, L, \lambda : P \to L) \\ Mor_{\mathbf{C}}((PS_1, L, \lambda_1), (PS_2, L, \lambda_2)) &= \{ \quad (f_{PS} : PS_1 \to PS_2, id_L : L \to L) \\ &\qquad\qquad |id_L \circ \lambda_1 = \lambda_2 \circ f_{P_1} \qquad \} \\ (f_{PS_1}, id_L) \circ (f_{PS_2}, id_L) &= (f_{PS_1} \circ f_{PS_2}, id_L \circ id_L) \\ id_{(PS,L,\lambda:P\to L)} &= (id_{PS}, id_L) \end{aligned}$$

Obviously, $\mathbf{C}$ is isomorphic to $\mathbf{PTSys}(\mathbf{L})$. Hence, $\mathbf{PTSys}(\mathbf{L})$ is a well-defined category.
The fact that $\mathbf{PTSys}(\mathbf{L})$ is a weak adhesive HLR category remains to be shown. Therefore, according to Theorem 4.15 (4.) in [EEPT06], the following statements have to be proven.

1. $(\mathbf{L}, \mathcal{M}_2)$ is a (weak) adhesive HLR category, which obviously holds for the class of all $\mathbf{L}$-morphisms $\mathcal{M}_2$

2. F preserves pushouts along $\mathcal{M}_{strict}$-morphisms, which is also obvious since pushouts in **PTSys** are constructed componentwise (see Fact A.19)

3. $Incl$ preserves pullbacks along $\mathcal{M}_2$ morphisms, which is obvious as well.

Therefore, $(\mathbf{C}, \mathcal{M}_{strict} \times \mathcal{M}_2)$ and for this reason also $(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}_{strict})$ are weak adhesive HLR categories. □

Obviously, a forgetful functor mapping L-labeled P/T systems to the corresponding unlabeled P/T systems exists. Note this functor is called *forgetful* since the functor "forgets" the labeling. This functor is required for some of the proofs in the next section.

Definition 5.4 (Forgetful Functor $V : \mathbf{PTSys}(\mathbf{L}) \to \mathbf{PTSys}$)**.** Forgetful functor $V : \mathbf{PTSys}(\mathbf{L}) \to \mathbf{PTSys}$ is defined by

$$V_{Ob}(PS, \lambda) = PS$$
$$V_{Mor}(f) = f$$

Finally, a reconfigurable labeled P/T system is defined formally. The definition is analogous to the definition in the case of P/T systems and AHL systems.

Definition 5.5 (Reconfigurable L-Labeled P/T System)**.** Given a L-labeled P/T system PSL and a set $RULES$ of rules, a reconfigurable (L-)labeled P/T system is defined by $(PSL, RULES)$.

5.2 Labeled P/T Systems as weak Adhesive HLR Category with NACs

In this section is shown, that the category $\mathbf{PTSys}(\mathbf{L})$ is a weak adhesive HLR category with NACs.

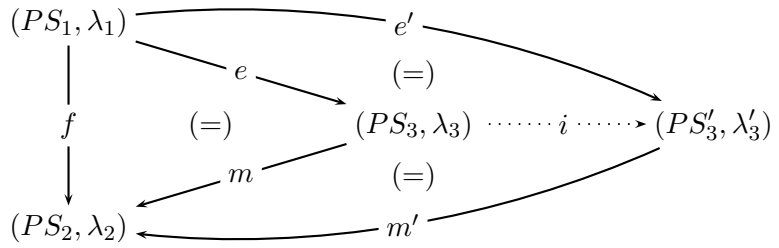
Theorem 5.6 ($(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is a Weak Adhesive HLR Category with NACs)**.** *The category* $(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ *with the following morphism classes is a weak adhesive HLR category with NACs:*

$\mathcal{M}$	:	*strict L-labeled P/T morphisms (see Definition 3.10)*
$\mathcal{M}'$	:	*injective L-labeled P/T morphisms (see Definition 3.8 and Fact C.1)*
$\mathcal{Q}$	:	*injective L-labeled P/T morphisms (see Definition 3.8 and Fact C.1)*
$\mathcal{E}'$	:	*minimal jointly surjective L-labeled P/T morphisms (see Definition 3.14)*

Proof. As already proven in Fact 5.3, ($\mathbf{PTSys}(\mathbf{L})$,$\mathcal{M}$) is a weak adhesive HLR category. Therefore, merely the additional properties of a *weak adhesive HLR category with NACs* (see Definition 2.4) need to be proven. These properties are proven in the following Facts 5.7, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.17 and 5.18. □

Fact 5.7 ($(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Unique Epi-$\mathcal{M}$ Factorization)**.** See Definition 2.5.

Proof. This is analogous to the construction in **PTSys** (see Fact 3.17) since Fact C.1 holds.



Choose $\lambda_3(e_P(p_1)) := \lambda_1(p_1)$ for all $p_1 \in P_1$. First, the well-definedness of $\mathbf{PTSys}(\mathbf{L})$-morphisms e and m is shown. By construction, this follows directly for e. Remember that m is an inclusion.

$$\forall p_3 \in P_3 \subseteq P_2 : \lambda_2(m_P(p_3)) = \lambda_2(p_3) = \lambda_2(f_P(p_1)) = \lambda_1(p_1) = \lambda_3(e_P(p_1)) = \lambda_3(p_3)$$

with $f_P(p_1) = p_3 = e_P(p_1)$.
The proof for the universal property is analogous to the case without labels (see Fact 3.17). The fact that $i : (PS_3, \lambda_3) \to (PS_3', \lambda_3')$ is a well-defined $\mathbf{PTSys}(\mathbf{L})$-morphism remains to be shown.

$$\forall p_3 \in P_3 \subseteq P_2 : \lambda_3(p_3) = \lambda_2(p_3) = \lambda_3'(m_P'^{-1}(p_3)) = \lambda_3'(i_P(p_3))$$

where m is an inclusion, m' is a monomorphism and $m' \circ i = m$ holds. □

Analogous to the case without labels, a unique $\mathcal{E}_0$-$\mathcal{M}'$ factorization is introduced in the next fact. It is used for proving that $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has unique $\mathcal{E}'$-$\mathcal{M}'$ pair factorization.

Fact 5.8 ($(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Unique $\mathcal{E}_0$-$\mathcal{M}'$ Factorization)**.** Let $\mathcal{E}_0$ be the class of minimal surjective L-labeled P/T morphisms (see Definition 3.13).

Proof. This follows directly since $(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has unique epi-$\mathcal{M}$ factorization (see Fact 5.7), $\mathcal{E}_0$-$\mathcal{M}'$ factorization and epi-$\mathcal{M}$ factorization only differ in the marking of PS_3 and $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has $\mathcal{E}_0$-$\mathcal{M}'$ fatorization (see Fact 3.18). Note that Fact C.1 holds. □

Fact 5.9 ($(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Unique $\mathcal{E}'$-$\mathcal{M}'$ Pair Factorization)**.** See Definition 2.6.

Proof. This is analogous to the proof of Fact 3.19 since $(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has unique $\mathcal{E}_0$-$\mathcal{M}'$ factorization and binary coproducts corresponding to binary coproducts in **PTSys** (see Fact C.3). Note jointly epimorphic morphisms in $\mathbf{PTSys}(\mathbf{L})$ are defined as in **PTSys** (see Fact C.2). □

Fact 5.10 $((\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has $\mathcal{M}$-$\mathcal{M}'$ Pushout-Pullback Decomposition Property**.** See Definition 2.7.

Proof. This fact holds since

1. the corresponding fact holds for $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ (see Fact 3.22),
2. pushouts (resp. pullbacks) along $\mathcal{M}$-morphisms in $\mathbf{PTSys}(\mathbf{L})$ are pushouts (resp. pullbacks) along $\mathcal{M}$-morphisms in $\mathbf{PTSys}$ (see Facts C.5 and C.6) and
3. $\mathcal{M}$- and $\mathcal{M}'$-morphisms are preserved by forgetful functor V (see Definition 5.4 and Fact C.1).

□

Fact 5.11 $((\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has $\mathcal{M}$-$\mathcal{Q}$ Pushout-Pullback Decomposition Property**.** See Definition 2.8. Analogous to 5.10 since $\mathcal{Q}$=$\mathcal{M}'$.

Fact 5.12 $((\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Initial Pushouts over $\mathcal{M}'$-Morphisms**.** See Definition 2.10.

Construction. The construction is analogous to the construction of initial pushouts in $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ (see Fact 3.25) with the following labeling functions:

$$\lambda_B = \lambda_{0_{|P_B}}$$
$$\lambda_C = \lambda_{1_{|P_C}}$$

$$\begin{array}{ccc} (PS_B, \lambda_B) & \xrightarrow{b} & (PS_0, \lambda_0) \\ \downarrow{\scriptstyle f'} & (1) & \downarrow{\scriptstyle f} \\ (PS_C, \lambda_C) & \xrightarrow{c} & (PS_1, \lambda_1) \end{array} \qquad \begin{array}{ccccc} (PS_B, \lambda_B) & \overset{b^*}{\dashrightarrow} & (PS_2, \lambda_2) & \xrightarrow{b'} & (PS_0, \lambda_0) \\ \downarrow{\scriptstyle f'} & (3) & \downarrow{\scriptstyle f''} & (2) & \downarrow{\scriptstyle f} \\ (PS_C, \lambda_C) & \underset{c^*}{\dashrightarrow} & (PS_3, \lambda_3) & \xrightarrow{c'} & (PS_1, \lambda_1) \end{array}$$

Proof. Remember that pushouts along $\mathcal{M}$-morphisms in $\mathbf{PTSys}(\mathbf{L})$ are pushouts along $\mathcal{M}$-morphisms in $\mathbf{PTSys}$ (see Fact C.5) and $\mathcal{M}$- and $\mathcal{M}'$-morphisms are preserved by forgetful functor V (see Definition 5.4 and Fact C.1).
Referring to the proof of Fact 3.25, the fact that all constructed morphisms are well-defined L-labeled P/T morphisms remains to be shown. This follows directly by construction for inclusions b and c.
Well-definedness of f':

$$\begin{aligned} \forall p_B \in P_B : \lambda_B(p_B) &= \lambda_0(b_P(p_B)) \\ &= \lambda_1(f_P(b_P(p_B))) \\ &= \lambda_1(c_P(f'_P(p_B))) \\ &= \lambda_C(f'_P(p_B)) \end{aligned}$$

Well-definedness of b^*:

$$\begin{aligned}\forall p_B \in P_B : \lambda_B(p_B) &= \lambda_0(b_P(p_B)) \\ &= \lambda_0(b'_P(b^*_P(p_B))) \\ &= \lambda_2(b^*_P(p_B))\end{aligned}$$

Well-definedness of c^*:

$$\begin{aligned}\forall p_C \in P_C : \lambda_C(p_C) &= \lambda_1(c_P(p_C)) \\ &= \lambda_1(c'_P(c^*_P(p_C))) \\ &= \lambda_3(c^*_P(p_C))\end{aligned}$$

Recapitulating, (1) and (3) are pushouts in $\mathbf{PTSys}(\mathbf{L})$ since they are pushouts in **PTSys** (see Fact C.5) and (2) is constructed as pushout in $\mathbf{PTSys}(\mathbf{L})$. Hence, (1) is the initial pushout over f' in $(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$. □

Fact 5.13 ($\mathcal{M}'$ in $(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is closed under POs and PBs along $\mathcal{M}$-Morphisms)**.** See Definition 2.11.

Proof. Analogous to the proof of the corresponding fact in **PTSys** (see Fact 3.26) since Fact C.1 holds. □

Fact 5.14 ($\mathcal{Q}$ in $(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is closed under POs and PBs along $\mathcal{M}$-Morphisms)**.** See Definition 2.12. Analogous to Fact 5.13 since $\mathcal{Q}{=}\mathcal{M}'$.

Fact 5.15 ($(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Induced Pullback-Pushout Property for $\mathcal{M}$ and $\mathcal{Q}$)**.** See Definition 2.13.

Proof. Trivial, since $(\mathbf{PTSys}, \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has this property (see Fact 3.28) and monomorphisms, pushouts and pullbacks along $\mathcal{M}$-morphisms in $\mathbf{PTSys}(\mathbf{L})$ are defined as in **PTSys** (see Facts C.1, C.5 and C.6). □

Fact 5.16 ($(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Composition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$)**.** See Definition 2.14.

Proof. Let $f : A \to B \in \mathcal{Q}$ and $g : B \to C \in \mathcal{M}'$ be $\mathbf{PTSys}(\mathbf{L})$-morphisms. From standard category theory follows that $g \circ f \in \mathcal{Q}$ (since the composition of monomorphisms is monomorphism). □

Fact 5.17 ($(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ has Decomposition Property for Morphisms in $\mathcal{M}'$ and $\mathcal{Q}$)**.** See Definition 2.15.

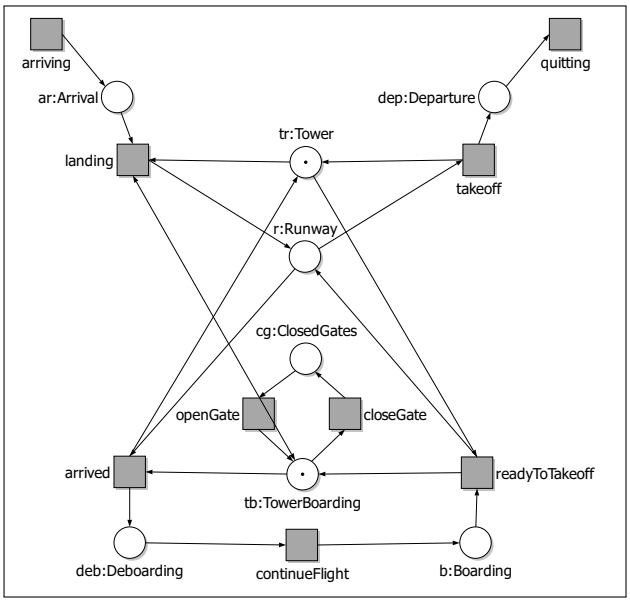
Proof. Let $g \circ f \in \mathcal{Q}$ and $g \in \mathcal{M}'$ be $\mathbf{PTSys}(\mathbf{L})$-morphisms. From standard category theory follows that $f \in \mathcal{Q}$ (decomposition property of monomorphisms). □

Fact 5.18 ($\mathcal{Q}$ in $(\mathbf{PTSys}(\mathbf{L}), \mathcal{M}, \mathcal{M}', \mathcal{E}', \mathcal{Q})$ is closed under Composition and Decomposition)**.** See Definition 2.16. Analogous to Fact 5.16 and Fact 5.17 since $\mathcal{Q}{=}\mathcal{M}'$.

5.3 Case Study: Airport Control System with Labels

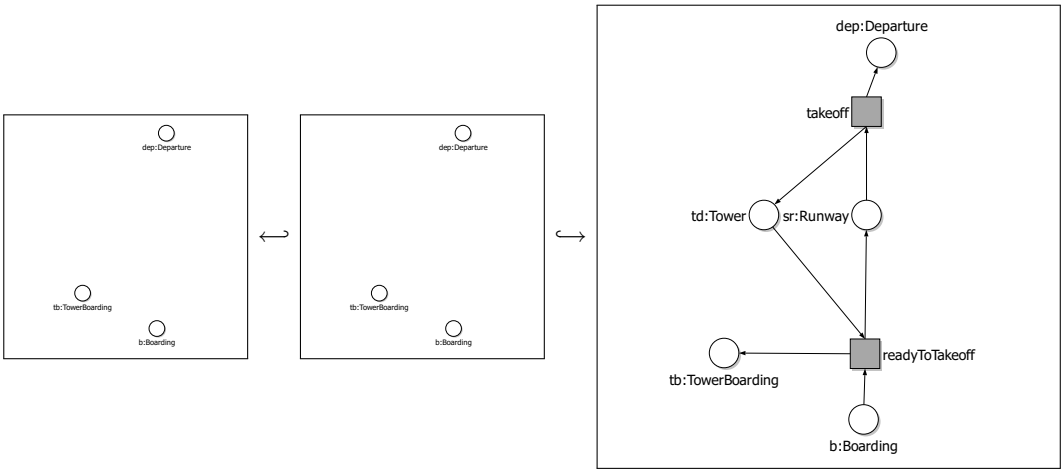
This section illustrates the advantages of labeled P/T systems. Therefore, the startsystem and some rules of the airport control system (ACS) introduced in Section 3.3 are expanded to labeled P/T systems. Note that this section does not comprise a self-contained example.
In this example, the usual notation for labels is used: Label l of place p is denoted by $p : l$, which means $\lambda(p) = l \in L$.

Startsystem: $LLvl1ap$



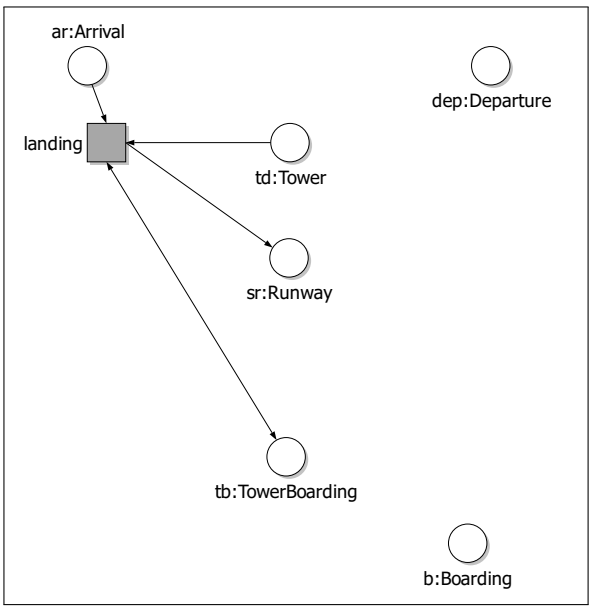
This is the labeled startsystem of this example. It corresponds to the startsystem of ACS (see startsystem in Subsection 3.3.3), where the names of the places are used as labels.

Rule 3b: *addStartingRunway*



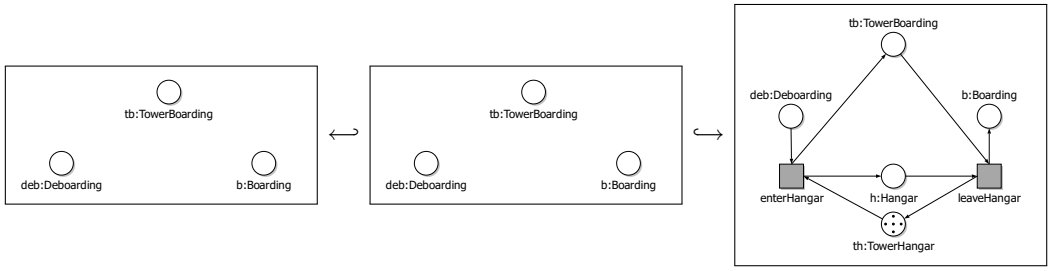
Adding a starting runway to a level-2-airport is expressed by this rule. It is smaller and, hence, easier to overlook than the corresponding rule in unlabeled ACS (see rule 3b in Subsection 3.3.3). Likewise this rule corresponds rather to the usual modeling approach and is to be understood intuitively. The left-hand side does not contain a runway since nonsensical matches are excluded by labeling.

Rule 3b - NAC 1:



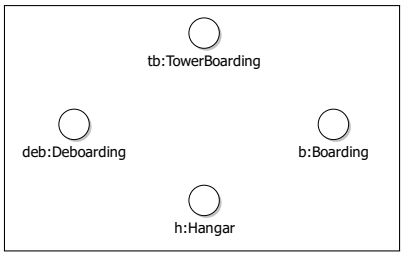
Analogous to the negative application condition of the corresponding rule in unlabeled ACS (see rule 3b in Subsection 3.3.3), the application of this rule to a level-1-airport is forbidden by this negative application condition. The existence of the edges between the transition *landing* and the place $tb : TowerBoarding$ is a sufficient condition for a level-1-airport.

Rule 7a: *addHangarWithoutExistingMaintenance*



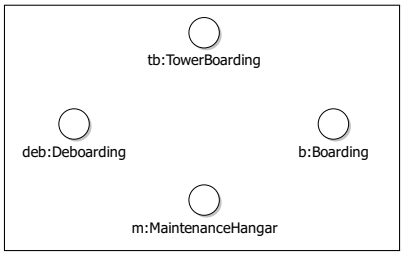
The rule for adding a hangar under the assumption that no maintenance hangar exists yet is much smaller than the corresponding unlabeled rule (see rule 7a in Subsection 3.3.3) since unsensical matches are excluded by labeling. Additionally, it is to be understood intuitively in contrast to the corresponding unlabeled rule.

Rule 7a - NAC 1:



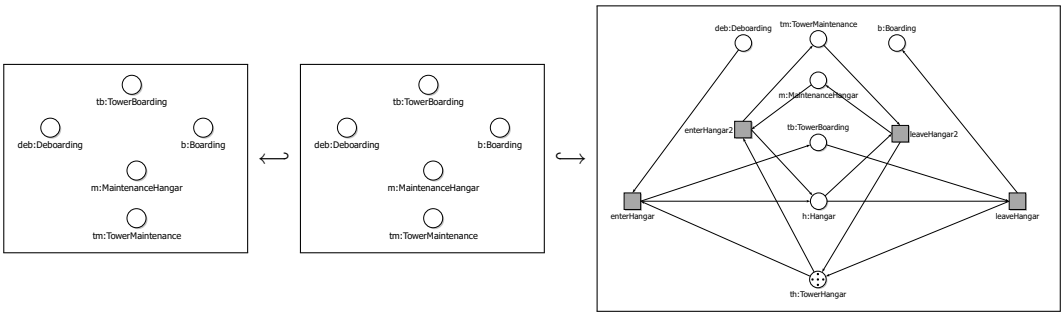
This negative application condition prevents this rule from being applied repeatedly. Comparing this NAC with the corresponding unlabeled NAC (see NAC 1 of rule 7a in Subsection 3.3.3) shows the advantages of the labels clearly. The unlabeled NAC is, in contrast to this NAC, voluminous and very hard to understand. Although, the condition that this rule can only be applied if no maintenance hangar exists is not ensured by this negative application condition. Therefore, an additional negative application condition is required.

Rule 7a - NAC 2:



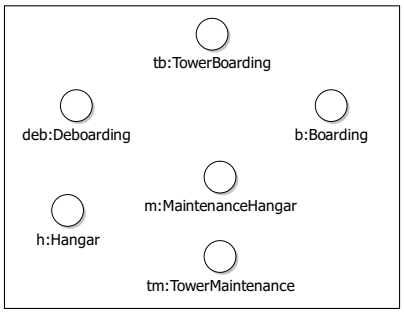
Through this negative application condition, the rule can only be applied if no maintenance hangar exists at the airport.

Rule 7b: $addHangarWithExistingMaintenance$



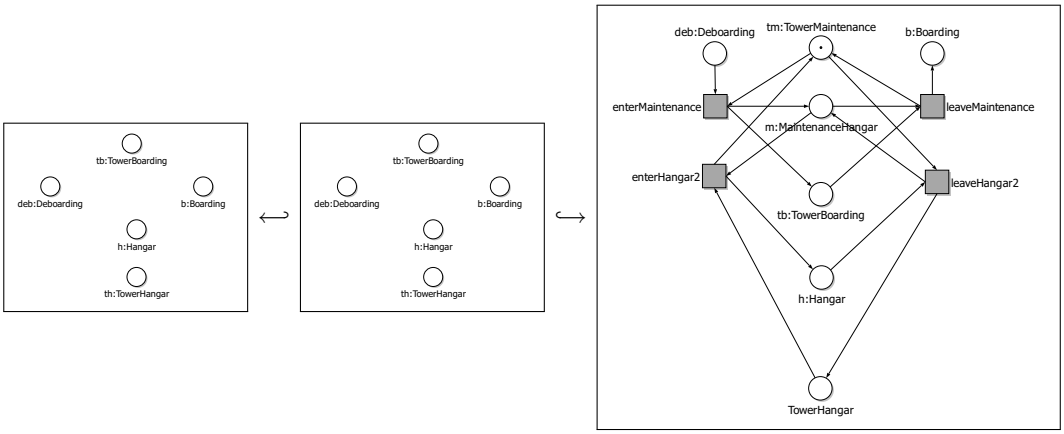
Adding a hangar under the assumption that a maintenance hangar already exists is modeled by this rule. In addition to the fact that the rule is clearly smaller than the suitable rule without NACs (see rule 7b in Subsection 3.3.3), this rule requires only one negative application condition. This is because a match from the maintenance hangar to a hangar is not possible because of the labeling.
This example illustrates that the modeling with labels is less error-prone than without.

Rule 7b - NAC 1:



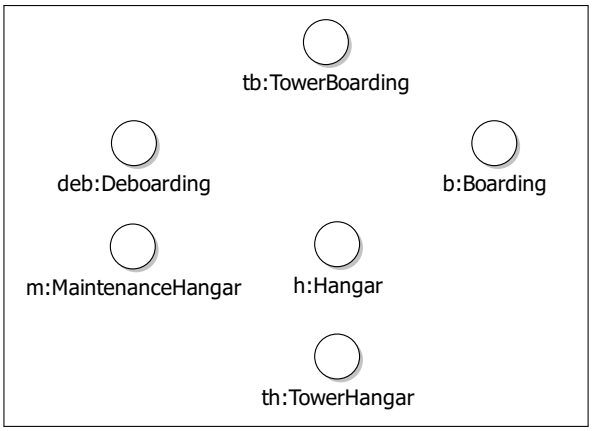
This negative application condition prevents this rule from being applied repeatedly.

Rule 8b: *addMaintenanceWithExistingHangar*



The use of labels leads to the possibility to express the adding of a maintenance hangar under the assumption that a hangar already exists in one rule. In the case of unlabeled ACS (see rule 8b and 8c in Subsection 3.3.3) two rules are required to prevent mappings from the hangar to a maintenance hangar since the only difference between hangar and maintenance hangar is the token count. Another advantage of this rule is that it is smaller than the suitable rules in unlabeled ACS (see rule 8b and 8c in Subsection 3.3.3).

Rule 8b - NAC 1:



This negative application condition prevents this rule from being applied repeatedly.

Chapter 6

Conclusion

6.1 Summary

In this thesis, reconfigurable Petri systems are extended by negative application conditions. Therefore, it is shown that the related categories are weak adhesive HLR categories with NACs. These proofs are accomplished for the case of (non-labeled) P/T systems, AHL nets, AHL systems and L-labeled P/T systems. In this context, the analyzed categories are introduced and their special morphisms and categorical constructions are identified and formally proven.
Weak adhesive HLR categories with NACs are weak adhesive HLR categories with three additional distinguished morphism classes and some special properties. Table 6.1 gives an overview of the achieved results for Petri nets and Petri systems. Note that the proofs for P/T nets and L-labeled P/T nets are not accomplished explicitly in this thesis. Although, these results can be derived from the corresponding proofs for P/T systems and L-labeled P/T systems. Note also that L-labeled P/T systems in this thesis are P/T systems with labeled places and unlabeled transitions.

	nets	systems
P/T	$\mathcal{M}$: injective $\mathcal{M}'$: injective $\mathcal{Q}$: injective $\mathcal{E}'$: jointly surjective	$\mathcal{M}$: strict $\mathcal{M}'$: injective $\mathcal{Q}$: injective $\mathcal{E}'$: minimal jointly surjective
labeled P/T	$\mathcal{M}$: injective $\mathcal{M}'$: injective $\mathcal{Q}$: injective $\mathcal{E}'$: jointly surjective	$\mathcal{M}$: strict $\mathcal{M}'$: injective $\mathcal{Q}$: injective $\mathcal{E}'$: minimal jointly surjective
AHL	$\mathcal{M}$: strict injective $\mathcal{M}'$: injective $\mathcal{Q}$: injective $\mathcal{E}'$: jointly equation strict and jointly surjective	$\mathcal{M}$: strict $\mathcal{M}'$: injective $\mathcal{Q}$: injective $\mathcal{E}'$: jointly equation strict and minimal jointly surjective

Table 6.1: Requirements for Petri Nets and Systems to be weak adhesive HLR categories with NACs

$\mathcal{Q}$-morphisms influence the satisfaction of negative application conditions directly. Hence, it is a very important result that the morphism class $\mathcal{Q}$ is the class of all monomorphisms in every analyzed category. This is especially significant for the categories of Petri systems (**PTSys**, **AHLSystems** and **PTSys**(**L**)).
A restriction of $\mathcal{Q}$-morphisms to marking strict morphisms would limit the significance of negative application conditions for Petri systems extremely. In the case of such a restriction, one negative application condition would be required for every possible marking of the net. This would cause an infinite set of negative application conditions in most cases and, therefore, a NAC would have less expressive power than in the case with $\mathcal{Q}$ being the class of all monomorphisms.
With the obtained results (see Table 6.1), negative application conditions can be used in reconfigurable Petri systems without losing important qualities like the *Local Church-Rosser Theorem*, the *Parallelism Theorem*, the *Completeness Theorem of Critical Pairs*, the *Concurrency Theorem*, the *Embedding* and the *Extension Theorem* and the main result the *Local Confluence Theorem*. All these theorems are lifted up for the use of negative application conditions in [Lam07] and [LEOP08] and are presented in this thesis as well.
An interesting side result of the achieved results (see Table 6.1) is that the *Concurrency Theorem*, the *Embedding* and the *Extension Theorem*, the *Completeness Theorem of Critical Pairs* and also the *Local Confluence Theorem* can now be used even in the case without NACs in all analyzed categories. This is because these theorems require some of the proven properties, e.g. initial pushouts over $\mathcal{M}'$-morphisms are required for the application of the *Embedding* and the *Extension Theorem*.
The practical applicability of the theoretical results is demonstrated with two small case studies. An airport control system, called ACS, is presented in form of a reconfigurable P/T system with NACs. ACS is designed to prevent accidents at the airport. It secures that some areas of the airport, like the starting or the landing runway, can only be used exclusively by a given amount of airplanes. ACS can adapt to various changes of the airport, e.g. adding and removing of starting and landing runways, adding and removing additional gates, adding a hangar and adding a maintenance hangar. Later, ACS is extended to AHL-ACS, a reconfigurable AHL system with NACs with additional functionalities. In contrast to ACS, AHL-ACS is responsible for the coordination of the airplanes at the gates. In this example, every airplane and every gate has a determined size. An airplane is only allowed to use a specific gate if it is of the same size. The size of airplanes and gates is modeled as data type. Additional sizes can be added and removed, which is represented by an extension of the data type. Gates of arbitrary sizes can also be added and removed, which is expressed by changing the net structure.
The case studies demonstrate the necessity of negative application conditions for expressing several conditions. In the AHL case, the specification is not fixed, in contrast to most common examples. Some of the mentioned theorems above are applied to the examples for showing their practical relevance and demonstrating their significance.
Finally, ACS is exemplarily converted to a L-labeled P/T system and the advantages

of L-labeled P/T systems are analyzed. Labels are extremely useful for ensuring intuitive modeling since their use prevents most nonsensical matches. Thus, modeling with labels is less error-prone and the rules are in general much smaller and easier to overlook than the corresponding rules without labels.

6.2 Future Work

One ongoing research task are algebraic higher order (AHO) nets (see [HME05] and [Hof02]), which allow the existence of dynamical tokens like Petri systems and transformation rules. They can be considered as AHL nets with a higher-order specification. In the ongoing research project *forMAlNET* (see [for08]) of the *TFS-group* at *Technische Universität Berlin*, these nets are used for modeling workflows of mobile ad-hoc networks. In this context, the AHO nets have a specific (higher-order) specification and algebra formalizing Petri systems with firing steps, rules and the application of rules to the Petri systems and various features. These nets have two kinds of places. On the one hand there are *system places* containing tokens of the sort *system*, i.e. Petri systems, and on the other hand there are *rule places* containing tokens of the sort *rule*, i.e. rules for transforming Petri systems. By firing transitions on the *system level* of the AHO net, the Petri systems on the *token level* can be moved, fired and dynamically changed. Up to now, negative application conditions cannot be used within AHO nets. Therefore, the NACs have to be integrated into the undelying algebra of AHO nets.
Another ongoing research is the extension of the results of this thesis to *open AHL nets* and *open AHL systems*. They are developed in the case of the diploma thesis of Conny Ullrich (see [Ull08]) and are an extension of AHL nets by *open places* and *communication transitions*. In contrast to AHL nets, an open AHL has three separate sets of places - *local*, *input* and *output places* - and two sets of transitions - *local* and *communication transitions*. Arbitrary tokens can appear at input places and arbitrary tokens can disappear from output places. Thus, open places are suitable for modeling communication channels and for expressing external events. The categories **OAHLNets** of open AHL nets and **OAHLSystems** of open AHL systems are introduced in [Ull08] and the fact that they are weak adhesive HLR categories is proven. Hence, transformation rules for open AHL nets and systems can be formulated. Moreover, the identification of input and output places by transformations is possible. This is feasible if, for example, an external event should be changed to an internal event or if communication is modeled by open places and the current communication partner is an actor of the AHL system itself. Up to now, using negative application conditions is not possible in reconfigurable open AHL nets and systems without losing important qualities of adhesive HLR systems. Therefore, the additional properties of adhesive HLR categories with NACs have to be proven for the categories **OAHLNets** of open AHL nets and **OAHLSystems** of open AHL systems.
With regard to the mentioned research project *forMAlNET* (see [for08]) which fo-

cuses on the formal modeling of mobile ad-hoc networks with algebraic higher-order nets, a tool to support visual editing and simulation of AHO nets (resp. systems) was developed within the *Visual Languages Project* at the *Technische Universität Berlin* (see [RON08]). Up to now, this editor, developed as an Eclipse-Plugin (see [ECL08]), is restricted to *reconfigurable object nets* (RONs) and can only handle injective mappings. RONs are a subclass of AHO nets with fixed transition types and do not have a formal semantic given by a signature and algebra. Nevertheless, RONs are sufficient for modeling the most examples of MANETs. Negative application conditions can be used within the Ron-Editor (see [RON08]), although, only injective morphisms are supported. The Ron-Editor is an ongoing project and still under development. In this context, the extension of the editor to general morphisms and the theory of adhesive HLR systems with NACs is a reasonable upgrade.

Appendix A

P/T Nets and P/T Systems

A.1 The Category of P/T Nets

The most important concepts of the category **PTNet**, with respect to this thesis, are introduced in this section and their correctness is proven.

A.1.1 Special Morphisms

In this subsection, special morphisms of the category **PTNet** are introduced. Therefore, special morphisms in **Sets** are presented and the correctness of these constructions is proven. The definitions of the special morphisms in **Sets** are used in the proofs of the special morphisms in **PTNet**.

Fact A.1 (Monomorphisms in **Sets**)**.** Injective functions are monomorphisms in category **Sets**.

Proof. **Part 1** ($\Rightarrow$). Let $m : B \to C$ be an injective function and $f, g : A \to B$ be functions with $m \circ f = m \circ g$.

$$\begin{array}{lrcll} & \forall a \in A: & (m \circ f)(a) & = & (m \circ g)(a) & \\ \Rightarrow & \forall a \in A: & m(f(a)) & = & m(g(a)) & \\ \Rightarrow & \forall a \in A: & f(a) & = & g(a) & m \text{ injective} \\ \Rightarrow & & f & = & g & \end{array}$$

Part 2 ($\Leftarrow$). Let $m : B \to C$ be a monomorphism in **Sets**.
Suppose that m is not injective, i.e. $\exists b_1 \neq b_2 \in B : m(b_1) = m(b_2)$. There exist functions $f, g : A \to B$ with $f(a) = b_1$ and $g(a) = b_2$ with $a \in A$ and $\forall a' \neq a \in A : f(a') = g(a')$.

$$\begin{array}{lrcll} \Rightarrow & \forall a \in A: & m(f(a)) & = & m(g(a)) & \\ \Rightarrow & & m \circ f & = & m \circ g & \\ \Rightarrow & & f & = & g & m \text{ monomorphism} \end{array}$$

This is a contradiction to the definition of f and g. □

Fact A.2 (Monomorphisms in **PTNet**)**.** Injective P/T net morphisms are monomorphisms in category **PTNet**.

Proof. Let $PN_x = (P_x, T_x, pre_x, post_x)$ for $x = A, B, C$ be P/T nets.

$$PN_A \overset{f}{\underset{g}{\rightrightarrows}} PN_B \xrightarrow{m} PN_C$$

Part 1 ($\Rightarrow$). Let $m : PN_B \to PN_C$ be an injective P/T net morphism and $f, g : PN_A \to PN_B$ be P/T net morphisms with $m \circ f = m \circ g$.
From the definition follows that m_P and m_T are injective functions, i.e. m_P and m_T are monomorphisms in the category **Sets**, as proven above.

$$\begin{array}{rrcll} & m \circ f & = & m \circ g & \\ \Rightarrow & m_P \circ f_P & = & m_P \circ g_P & \\ & \wedge\ m_T \circ f_T & = & m_T \circ g_T & \\ \Rightarrow & f_P & = & g_P & m_P \text{ monomorphism} \\ & \wedge\ f_T & = & g_T & m_T \text{ monomorphism} \\ \Rightarrow & f & = & g & \end{array}$$

Part 2 ($\Leftarrow$). Let $m : PN_B \to PN_C$ be a monomorphism in **PTNet**.
In the following, the fact that m_P and m_T are monomorphisms in the category **Sets**, i.e. injective functions, is shown.

Suppose that m_P is not injective. Then $\exists p_1 \neq p_2 \in P_B : m_P(p_1) = m_P(p_2)$. Construct PN_A with

- $P_A = \{p_1\}$
- $T_A = \emptyset$
- $pre_A = \emptyset$
- $post_A = \emptyset$

and inclusion $f = (f_P, f_T) : PN_A \to PN_B$ and $g = (g_P, g_T) : PN_A \to PN_B$ with $g_P(p_1) = p_2$. The well-definedness of this construction is evident. Obviously, $m \circ f = m \circ g$ holds, although $f \neq g$. This is a contradiction to the fact that m is a monomorphism. Hence, m_P is injective.

Suppose that m_T is not injective. Then $\exists t_1 \neq t_2 \in T_B : m_T(t_1) = t_C = m_T(t_2)$. The well-definedness of m implies that $m_P^{\oplus}(pre_B(t_1)) = pre_C(m_T(t_1)) = pre_C(t_C) = pre_C(m_T(t_2)) = m_P^{\oplus}(pre_B(t_2))$ (*post* analogous). Since the fact that m is a monomorphism implies the injectivity of m_P, as already proven above, only the case with m_P being injective has to be considered in the following. So $pre_B(t_1) = pre_B(t_2)$ (*post* analogous). Construct PN_A with

- $P_A = \{p | p \in pre_B(t_1) \cup post_B(t_1)\}$.

- $T_A = \{t_1\}$
- $pre_A(t_1) = pre_B(t_1)$
- $post_A(t_1) = post_B(t_1)$

and inclusion $f = (f_P, f_T) : PN_A \to PN_B$ and $g = (g_P, g_T) : PN_A \to PN_B$ with inclusion $g_P(p) = p$ and $g_T(t_1) = t_2$. Well-definedness of PN_A and f is evident. Well-definedness of g:

- $pre_B(g_T(t_1)) = pre_B(t_2) = pre_B(t_1) = pre_A(t_1) = g_P^{\oplus}(pre_A(t_1))$
- *post* analogous.

Obviously, $m \circ f = m \circ g$ holds, although $f \neq g$. This is a contradiction to the fact that m is a monomorphism.
Hence, m_P and m_T are injective, i.e. m is injective. □

Fact A.3 (Epimorphisms in **Sets**)**.** Surjective functions are epimorphisms in category **Sets**.

Proof. **Part 1** ($\Rightarrow$). Let $e : A \to B$ be a surjective function and $f, g : B \to C$ be functions with $f \circ e = g \circ e$.

$$\begin{array}{lrcll} & \forall a \in A: & (f \circ e)(a) & = & (g \circ e)(a) & \\ \Rightarrow & \forall a \in A: & f(e(a)) & = & g(e(a)) & \\ \Rightarrow & \forall b \in B: & f(b) & = & g(b) & e \text{ surjective} \\ \Rightarrow & & f & = & g & \end{array}$$

Part 2 ($\Leftarrow$). Let $e : A \to B$ be an epimorphism in **Sets**.
Suppose that e is not surjective, i.e. $\exists b \in B : \nexists a \in A : e(a) = b$. There exist functions $f, g : B \to C$ with $f(b) \neq g(b)$ and $\forall b' \neq b \in B : f(b') = g(b')$.

$$\begin{array}{lrcll} \Rightarrow & \forall a \in A: & f(e(a)) & = & g(e(a)) & \\ \Rightarrow & & f \circ e & = & g \circ e & \\ \Rightarrow & & f & = & g & e \text{ epimorphism} \end{array}$$

This is a contradiction to the definition of f and g. □

Fact A.4 (Epimorphisms in **PTNet**)**.** Surjective P/T net morphisms are epimorphisms in category **PTNet**.

Proof. Let $PN_x = (P_x, T_x, pre_x, post_x)$ for $x = A, B, C$ be P/T nets.

$$PN_A \xrightarrow{\ e\ } PN_B \underset{g}{\overset{f}{\rightrightarrows}} PN_C$$

Part 1 ($\Rightarrow$). Let $e : PN_A \to PN_B$ be a surjective P/T net morphism and $f, g : PN_B \to PN_C$ be P/T net morphisms with $f \circ e = g \circ e$.
From the definition follows that e_P and e_T are surjective functions, i.e. e_P and e_T are epimorphisms in the category **Sets**, as proven above.

$$\begin{array}{rrcll} & f \circ e & = & g \circ e & \\ \Rightarrow & f_P \circ e_P & = & g_P \circ e_P & \\ \wedge & f_T \circ e_T & = & g_T \circ e_T & \\ \Rightarrow & f_P & = & g_P & e_P \text{ epimorphism} \\ \wedge & f_T & = & g_T & e_T \text{ epimorphism} \\ \Rightarrow & f & = & g & \end{array}$$

Part 2 ($\Leftarrow$). Let $e : PN_A \to PN_B$ be an epimorphism in **PTNet**.
In the following, the fact that e_P and e_T are epimorphisms in the category **Sets**, i.e. surjective functions, is shown.

Suppose e_T is not surjective. Then $\exists t_B \in T_B : \nexists t_A \in T_A : e_T(t_A) = t_B$. Construct PN_C with

- $P_C = P_B$
- $T_C = T_B \cup \{t'_b\}$ with $t'_B \notin T_B$.
- $pre_C(t) = \begin{cases} pre_B(t) & |t \in T_B \\ pre_B(t_B) & |t = t'_B \end{cases}$
- $post_C(t) = \begin{cases} post_B(t) & |t \in T_B \\ post_B(t_B) & |t = t'_B \end{cases}$

and inclusion $f = (f_P, f_T) : PN_B \to PN_C$ and $g = (g_P, g_T) : PN_B \to PN_C$ with

- $g_P(p) = p$
- $g_T(t) = \begin{cases} t'_B & |t = t_B \\ t & |else \end{cases}$

PN_C is a copy of the net PN_B, where transition t_B is duplicated. Well-definedness of PN_C and f is evident. Well-definedness of g:

- $\forall t \in T_B \backslash \{t_B\} : pre_C(g_T(t)) = pre_C(t) = pre_B(t) = g_P^{\oplus}(pre_B(t))$
- for $t = t_B : pre_C(g_T(t_B)) = pre_C(t'_B) = pre_B(t_B) = g_P^{\oplus}(pre_B(t_B))$
- *post* analogous.

Obviously, $f \circ e = g \circ e$ holds, although $f \neq g$. This is a contradiction to the fact that e is an epimorphism.

Suppose e_P is not surjective. Then $\exists p_B \in P_B : \nexists p_A \in P_A : e_P(p_A) = p_B$. Since the fact that e is an epimorphism implies the surjectivity of e_T, as already proven above, only the case with e_T being surjective has to be considered in the following. So no adjacent transition with p_B exists. Construct PN_C with

- $P_C = P_B \cup \{p'_B\}$ with $p'_B \notin P_B$.
- $T_C = T_B$.
- $pre_C = pre_B$
- $post_C = post_B$

and inclusion $f = (f_P, f_T) : PN_B \to PN_C$ and $g = (g_P, g_T) : PN_B \to PN_C$ with

- $g_P(p) = \begin{cases} p'_B & |p = p_B \\ p & |else \end{cases}$
- $g_T(t) = t$

PN_C is a copy of the net PN_B, where p_B is duplicated. Well-definedness of this construction is evident since p_B has no adjacent transitions.
Obviously, $f \circ e = g \circ e$ holds, although $f \neq g$. This is a contradiction to the fact that e is an epimorphism.
So e_P and e_T are surjective, i.e. e is surjective. □

Fact A.5 (Jointly Epimorphic **PTNet**-Morphisms)**.** Jointly surjective **PTNet**-morphisms are jointly epimorphic morphisms in category **PTNet**.

Proof. Analogous to the proof for epimorphic **PTNet**-morphisms (see Fact A.4). □

Fact A.6 (Isomorphisms in **Sets**)**.** Bijective functions are isomorphisms in category **Sets**.

Proof. **Part 1** ($\Rightarrow$). Let $i : A \to B$ be a bijective function.
A unique inverse function $i^{-1} : B \to A$ with the property $\forall b \in B : i^{-1}(b) = a$ if $i(a) = b$ exists. Obviously, $i^{-1} \circ i = id_A$ and $i \circ i^{-1} = id_B$.
Part 2 ($\Leftarrow$). Let $i : A \to B$ be an isomorphism in **Sets**.
The morphism hierarchy (standard category theory) implies that i is a monomorphism and an epimorphism. As already proven, this statement is equivalent to the statement that i is injective and surjective, i.e. i is bijective. □

Fact A.7 (Isomorphisms in **PTNet**)**.** Bijective P/T net morphisms are isomorphisms in category **PTNet**.

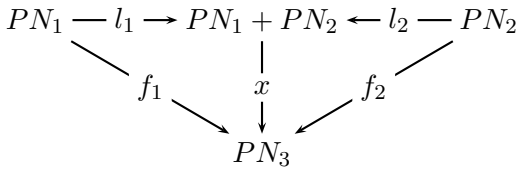
Proof. **Part 1** ($\Rightarrow$). Let $i : A \to B$ be a bijective P/T net morphism.
By definition, i_P and i_T are bijective functions. As proven in Fact A.6, i_P and i_T are isomorphisms in category **Sets**. So $\exists i_P^{-1} : i_P^{-1} \circ i_P = id_{A_P} \wedge i_P \circ i_P^{-1} = id_{B_P}$ and $\exists i_T^{-1} : i_T^{-1} \circ i_T = id_{A_T} \wedge i_T \circ i_T^{-1} = id_{B_T}$. Obviously, $i^{-1} = (i_P^{-1}, i_T^{-1})$ implies that $i^{-1} \circ i = id_A$ and $i \circ i^{-1} = id_B$.
Part 2 ($\Leftarrow$). Let $i : A \to B$ be an isomorphism in **PTNet**.
Obviously, i_P and i_T are isomorphisms in **Sets**. As already proven in Fact A.6, i_P and i_T are bijective functions, i.e. i is bijective. □

A.1.2 Categorical Constructions

In this subsection, binary coproducts, pushouts and pullbacks along monomorphisms in the category **PTNet** are presented. Since all these constructions are defined componentwise, the same constructions are also presented in the category **Sets**.

Fact A.8 (Binary Coproducts in **Sets**)**.** A set $A + B$ with two injective morphisms $i_1 : A \to A + B$ and $i_2 : B \to A + B$ is the binary coproduct of A and B in the category **Sets** if and only if $A + B$ is the disjoint union of A and B and i_1 and i_2 are injections mapping every element of the sets A and B to the corresponding object of the disjoint union.
This fact is well known and can be found in Example 27.4.2 in [EMC+01].

Fact A.9 (Binary Coproducts in **PTNet**)**.** A P/T net $PN_1 + PN_2 = (P_1 + P_2, T_1 + T_2, pre_{12}, post_{12})$ with two injective morphisms $l_1 : PN_1 \to PN_1 + PN_2$ and $l_2 : PN_2 \to PN_1 + PN_2$ is the binary coproduct of $PN_1 = (P_1, T_1, pre_1, post_1)$ and $PN_2 = (P_2, T_2, pre_2, post_2)$ in the category **PTNet** if and only if $P_1 + P_2$ is the disjoint union of P_1 and P_2, $T_1 + T_2$ is the disjoint union of T_1 and T_2, l_1 and l_2 are injections mapping every element of the sets P_1, P_2, T_1 and T_2 to the corresponding object of the disjoint union and pre_{12} and $post_{12}$ are induced by l_1 and l_2.



Proof. **Part 1** ($\Rightarrow$). Let $PN_1 + PN_2 = (P_1 + P_2, T_1 + T_2, pre_{12}, post_{12})$ with $P_1 + P_2 = P1 \uplus P_2$ and $T_1 + T_2 = T_1 \uplus T_2$ be a P/T net with pre_{12} and $post_{12}$ induced by $l_1 : PN_1 \to PN_1 + PN_2$ and $l_2 : PN_2 \to PN_1 + PN_2$. Consider P/T net $PN_3 = (P_3, T_3, pre_3, post_3)$ and morphisms $f : PN_1 \to PN_3$ and $g : PN_2 \to PN_3$. As proven in Fact A.8, $(P_1 + P_2, l_{1_P}, l_{2_P})$ is the coproduct in **Sets** of P_1 and P_2 and $(T_1 + T_2, l_{1_T}, l_{2_T})$ is the coproduct in **Sets** of T_1 and T_2. This fact implies the existence of morphisms $[f_P, g_P] : P_1 + P_2 \to P_3$ and $[f_T, g_T] : T_1 + T_2 \to T_3$ with $[f_P, g_P] \circ l_{1_P} = f_P$, $[f_P, g_P] \circ l_{2_P} = g_P$ and $[f_T, g_T] \circ l_{1_T} = f_T$ and $[f_T, g_T] \circ l_{2_T} = g_T$, which is equivalent to $[f, g] \circ l_1 = f$ and $[f, g] \circ l_2 = g$ for $[f, g] = ([f_P, g_P], [f_T, g_T])$. This morphism is well-defined since $PN_1 + PN_2$ is the componentwise disjoint union

of PN_1 and PN_2 and pre_{12} and $post_{12}$ are induced by the injections l_1 and l_2.
Part 2 ($\Leftarrow$). Let $(PN_1 + PN_2 = (P_1 + P_2, T_1 + T_2, pre_{12}, post_{12}), l_1 : PN_1 \to PN_1 + PN_2, l_2 : PN_2 \to PN_1 + PN_2)$ be the coproduct of $PN_1 = (P_1, T_1, pre_1, post_1)$ and $PN_2 = (P_2, T_2, pre_2, post_2)$ in **PTNet**.
In the next step, the facts that $(P_1 + P_2, l_{1_P}, l_{2_P})$ is the coproduct of P_1 and P_2 in **Sets** and $(T_1 + T_2, l_{1_T}, l_{2_T})$ is the coproduct of T_1 and T_2 in **Sets** are shown.

Suppose that $P_1 + P_2$ is not the disjoint union of P_1 and P_2. Several cases can be distinguished:

- $\exists p \in P_1 + P_2 \backslash (l_{1_P}(P_1) \cup l_{2_P}(P_2))$, i.e. (l_1, l_2) are not jointly surjective in **PTNet**. As already proven in Fact A.5, this is equivalent to the fact that (l_1, l_2) are not jointly epimorphic in category **PTNet**. This is a contradiction to Fact D.2.

- $\exists p \in l_{1_P}(P_1) \cup l_{2_P}(P_2)$. Choose $PN_3 = (P_1 \uplus P_2, T_1 \uplus T_2, pre_3, post_3)$, with

  - $pre_3(x) = \begin{cases} pre_1(x) & |x \in P_1 \\ pre_2(x) & |x \in P_2 \end{cases}$
  - *post* analogous.

  and inclusions $f : PN_1 \to PN_3$ (mapping to PN_1-part of PN_3) and $g : PN_2 \to PN_3$ (mapping to PN_2-part of PN_3). Well-definedness of this construction is evident. Since $p \in l_{1_P}(P_1) \cup l_{2_P}(P_2)$, but $f_P(p) \neq g_P(p)$ by construction, no unique morphism $x : PN_1 + PN_2 \to PN_3$ with $x \circ l_1 = f$ and $x \circ l_2 = g$ exists, which is a contradiction to the fact that $PN_1 + PN_2$ is the coproduct in **PTNet**.

- $\exists p \neq p' \in P_1 + P_2 : l_{1_P}(p) = l_{1_P}(p')$. Choose (PN_3, f_1, f_2) as componentwise disjoint union of PN_1 and PN_2 and $f_j : PN_j \to PN_3$ as inclusions for $j = 1, 2$ (as described above). Obviously, no morphism $x : PN_1 + PN_2 \to PN_3$ with $x \circ l_1 = f_1$ and $x \circ l_2 = f_2$ exists. This is a contradiction to the fact that $PN_1 + PN_2$ is the coproduct in **PTNet**. This is analogous for l_{2_P} being non-injective.

This is analogous for $T_1 + T_2$.
Therefore, $P_1 + P_2 = P_1 \uplus P_2$ and $T_1 + T_2 = T_1 \uplus T_2$ and $l_1 = (l_{1_P}, l_{1_T})$ and $l_2 = (l_{2_P}, l_{2_T})$ are injections mapping every element of the sets P_1, P_2, T_1 and T_2 to the corresponding element of the disjoint union. pre_{12} and $post_{12}$ are induced by l_1 and l_2. □

Remark A.10 (Compatibility of Coproducts with Injective P/T Net Morphisms). Coproducts in **PTNet** are compatible with injective P/T net morphisms (i.e. monomorphisms). This fact holds since coproducts in **PTNet** are componentwise in **Sets**. The compatibility of coproducts with monomorphisms (resp. $\mathcal{M}$-morphisms) is essential for the Parallelism Theorem. For a formal definition see Definition 5.14 in [EEPT06].

Fact A.11 (Pushouts in **Sets**)**.** In the category **Sets** the pushout of a span $C \xleftarrow{g} A \xrightarrow{f} B$ can be constructed by the quotient $D = B \uplus C_{/\equiv}$, where $\equiv$ is the smallest equivalence relation with $(f(a) = g(a)) \in \equiv$ for all $a \in A$. The morphisms $f' : C \to D$ and $g' : B \to D$ are defined by $f'(c) = [c]$ for all $c \in C$ and $g'(b) = [b]$ for all $b \in B$. Fact 2.17 in [EEPT06] contains the formal proof of this fact.

Fact A.12 (Pushouts in **PTNet**)**.** Pushouts in **PTNet** exist and are constructed componentwise in **Sets** for places and transitions (see Section 7.2 in [EP04]).
The formal proof for pushouts along $\mathcal{M}$-morphisms can be found in Fact 4.21 in [EEPT06]. This proof bases on the construction of pushouts in comma categories (see Definition A.41 in [EEPT06]). See also Fact 2 in [PEL07] for the formal proof of componentwise pushouts in comma categories.

Fact A.13 (Pullbacks in **Sets**)**.** In the category **Sets** the pullback of a span $C \xrightarrow{f} D \xleftarrow{g} B$ is given by $A = \bigcup_{d \in D} f^{-1}(d) \times g^{-1}(d) = \{(c, d) | f(c) = g(b)\} \subseteq C \times B$ with morphisms $f' : A \to B : (c, b) \mapsto b$ and $g' : A \to C : (c, b) \mapsto c$.
The proof is given in Fact 2.23 in [EEPT06].

Fact A.14 (Pullbacks in **PTNet** along Monomorphisms)**.** In the category **PTNet** pullbacks along monomorphisms can be constructed componentwise for places and transitions.

Proof-Idea. **PTNet** is isomorphic to the comma category (see Definition A.41 in [EEPT06]) $ComCat(ID_{\mathbf{Sets}}, \square^{\oplus}; \mathcal{I})$ with $\mathcal{I} = \{1, 2\}$, where $\square^{\oplus}$ is the free commutative monoid functor. According to Lemma A.40 in [EEPT06], $\square^{\oplus} : \mathbf{Sets} \to \mathbf{Sets}$ preserves pullbacks along monomorphisms. Hence, pullbacks along monomorphisms can be constructed componentwise. □

Remark A.15 (General Pullbacks in **PTNet**). The category **PTNet** has (general) pullbacks, but they cannot be constructed componentwise. See Fact A.24 in [EEPT06]. However, in this thesis only pullbacks along monomorphisms are considered.

A.2 The Category of P/T Systems

This section contains the most important concepts of the category **PTSys** with respect to this thesis.

A.2.1 Special Morphisms

Fact A.16 (Special Morphisms in **PTSys**)**.** Monomorphisms (resp. epimorphisms, jointly epimorphic morphisms) in **PTSys** are **PTSys**-morphisms that are monomorphisms (resp. epimorphisms, jointly epimorphic) in **PTNet**. Isomorphisms in **PTSys** are marking strict **PTSys**-morphisms which are isomorphisms in **PTNet**.
The proofs therefore are analogous to the corresponding proofs in **PTNet** (see

Facts A.2 for monomorphisms, A.4 for epimorphisms, A.5 for jointly epimorphic morphisms and A.7 for isomorphisms).

A.2.2 Categorical Constructions

In this subsection, the most important categorical constructions, with respect to this thesis, are introduced. Binary coproducts are required for the Parallelism Theorem and for some proofs in this thesis. The definitions of pushouts and pullbacks are required for every adhesive HLR system.

Fact A.17 (Binary Coproducts in **PTSys**)**.** The definition of binary coproducts in **PTSys** is the same as in **PTNet** with the additional assumption that the injections l_1 and l_2 are marking strict.

Proof. Analogous to the proof for **PTNet** (see Fact A.9). Obviously, the marking strictness of the injections is required to ensure the existence of the unique morphism for the universal property. □

Remark A.18 (Compatibility of Coproducts with Strict P/T Morphisms)*.* Coproducts in **PTSys** are compatible with strict P/T morphisms. This fact holds since coproducts in **PTNet** are constructed componentwise in **Sets** and coproduct inclusions are marking strict.

Fact A.19 (Pushouts in **PTSys** along Strict Morphisms)**.** Pushout $(PN_2, M_2) \overset{n}{\to} (PN_3, M_3) \overset{g}{\leftarrow} (PN_1, M_1)$ over the morphisms $m : (PN_0, M_0) \to (PN_1, M_1)$ and $f : (PN_0, M_0) \to (PN_2, M_2)$, where m is a strict P/T morphism (see Definition 3.10), can be constructed as pushout in **PTNet**. The marking M_3 is defined by

(1) $\forall p_1 \in P_1 \backslash m(P_0) : M_3(g(p_1)) = M_1(p_1)$

(2) $\forall p_2 \in P_2 \backslash f(P_0) : M_3(n(p_2)) = M_2(p_2)$

(3) $\forall p_0 \in P_0 : M_3(n \circ f(p_0)) = M_2(f(p_0))$

for P_i being the places of PN_i for $i = 0..3$. n is also a strict P/T morphism.
This fact is proven in Theorem 1 in [EEH+07].

Fact A.20 (Pullbacks in **PTSys** along Strict Morphisms)**.** Given $g : (PN_1, M_1) \to (PN_3, M_3)$ and $n : (PN_2, M_2) \to (PN_3, M_3)$, where n is a strict P/T morphism (see Definition 3.10), the pullback $(PN_2, M_2) \overset{f}{\leftarrow} (PN_0, M_0) \overset{m}{\to} (PN_1, M_1)$ in **PTSys** is constructed as pullback in **PTNet** and the marking M_0 is defined by

$$\forall p_0 \in P_0 : M_0(p_0) = M_1(m(p_0))$$

for P_0 being the places of PN_0. m is also a strict P/T morphism.
This fact is proven in Theorem 2 in [EEH+07].

A.2.3 Gluing Condition

For transforming P/T systems, a necessary and sufficient gluing condition (see Definition 2.18) for the applicability of rules to P/T systems is formalized in the next definition.

Fact A.21 (Gluing Condition for **PTSys**)**.** Let $\mathcal{M}$ be the class of strict **PTSys**-morphisms and $PS_x = (P_x, T_x, pre_x, post_x, M_x)$ for $x = 1, L, K, R$ be P/T systems. Given a production $p = (PS_L \overset{l \in \mathcal{M}}{\leftarrow} PS_K \overset{r \in \mathcal{M}}{\rightarrow} PS_R)$ in **PTSys** and a match $m : PS_L \to PS_1$, the gluing points GP, the dangling points DP and the identification points IP of PS_L are defined by

$$\begin{aligned} GP &= l(P_K \cup T_K), \\ DP &= \{p \in P_L | \exists t \in (T_1 \backslash m_T(T_L)) : m_P(p) \in pre_1(t) \oplus post_1(t)\} \\ IP &= \{p \in P_L | \exists p' \in P_L : p \neq p' \wedge m_P(p) = m_P(p')\} \cup \\ &\quad \{t \in T_L | \exists t' \in T_L : t \neq t' \wedge m_T(t) = m_T(t')\} \end{aligned}$$

The match m satisfies the gluing condition with respect to p if and only if the following statements hold:

1. $IP \subseteq GP$ (identification condition)
2. $DP \subseteq GP$ (dangling condition)
3. m is marking strict on places to be deleted, i.e. $\forall p \in P_L \backslash l(P_K) : M_L(p) = M_1(m(p))$

A rule in **PTSys** is applicable at a match m if and only if the gluing condition is satisfied for m (proof in [EEH+07] Section 5).

Appendix B

AHL Nets and AHL Systems

B.1 The Category of AHL Nets

In this section, the most important concepts of the category **AHLNet**, with respect to this thesis, are introduced.

B.1.1 Special Morphisms

Special morphisms of the category **AHLNet** are introduced in this subsection. Therefore, special morphisms in **Sig** and **Spec** are presented and the correctness of these constructions is proven. The definitions of the special morphisms in **Sig** and **Spec** are used in the proofs of the special morphisms in **AHLNet**.

Fact B.1 (Monomorphisms in **Sig**)**.** Injective signature morphisms are monomorphisms in category **Sig**.

Proof. Given a signature morphism $m = (m_S, m_{OP}) : (S_2, OP_2) \to (S_3, OP_3)$.
Part 1 ($\Rightarrow$). Let m be an injective signature morphism, i.e. m_S and m_{OP} are injective (monomorphisms in **Sets**), and $f, g : (S_1, OP_1) \to (S_2, OP_2)$ be signature morphisms with $m \circ f = m \circ g$.

$$\begin{array}{rrcll} & m \circ f & = & m \circ g & \\ \Rightarrow & m_x \circ f_x & = & m_x \circ g_x & \text{for } x = S, OP \\ \Rightarrow & f_x & = & g_x & m_x \text{ monomorphism for } x = S, OP \\ \Rightarrow & f & = & g & \end{array}$$

Part 2 ($\Leftarrow$). Let m be a monomorphism in category **Sig**.
In the following, the fact that m_S and m_{OP} are monomorphisms in category **Sets**, i.e. injective functions, is shown.

Suppose that m_S is not injective. Then $\exists s_2 \neq s_2' \in S_2 : m_S(s_2) = m_S(s_2')$ Construct (S_1, OP_1) with

- $S_1 = \{s_2\}$

- $OP_1 = \emptyset$

and inclusion $f : (S_1, OP_1) \to (S_2, OP_2)$ and $g = (g_S, g_{OP}) : (S_1, OP_1) \to (S_2, OP_2)$ with $g_S(s_2) = s_2'$ and $g_{OP} = \emptyset$.
Well-definedness of this construction is evident. Obviously, $m \circ f = m \circ g$ holds, although $f \neq g$. This is a contradiction to the fact that m is a monomorphism.

Suppose that m_{OP} is not injective. Then $\exists op_2 \neq op_2' \in OP_2 : m_{op}(op_2) = m_{op}(op_3)$. If op_2 and op_2' have different signatures, m_S is not injective. As already proven, this is a contradiction to the fact that m is a monomorphism.
If op_2 and op_2' have the same signature, construct (S_1, OP_1) with

- $S_1 = sorts(op_2)$, where $sorts(op_2)$ is the set of all sorts of the signature of op_2
- $OP_1 = \{op_2\}$

and inclusion $f : (S_1, OP_1) \to (S_2, OP_2)$ and $g = (g_S, g_{OP}) : (S_1, OP_1) \to (S_2, OP_2)$ with inclusion g_S and $g_{OP}(op_2) = op_2'$.
Well-definedness of this construction is evident. Obviously, $m \circ f = m \circ g$ holds, although $f \neq g$. This is a contradiction to the fact that m is a monomorphism. □

Fact B.2 (Monomorphisms in **Spec**)**.** Injective specification morphisms are monomorphisms in category **Spec**.

Proof. Analogous to the proof for monomorphisms in **Sig** (see Fact B.1). Choose $E_1 = \emptyset$ in part 2 of the mentioned proof. □

Fact B.3 (Monomorphisms in **AHLNet**)**.** Injective **AHLNet**-morphisms are monomorphisms in the category **AHLNet**.

Proof. Let $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2, 3$ be AHL nets.
Part 1 ($\Rightarrow$). Let $m = (m_{SP}, m_P, m_T, m_A) : AN_2 \to AN_3$ be an injective **AHLNet**-morphism and $f, g : AN_1 \to AN_2$ be **AHLNet**-morphisms with $m \circ f = m \circ g$.
By definition, m_P, m_T, m_S, m_{OP} and m_X are injective functions, i.e. monomorphisms in category **Sets**, as already proven in Fact A.1. Note that m_A is an isomorphism. The morphism hierarchy (standard category theory) leads to the fact that m_A is a monomorphism (and an epimorphism).

$$\begin{array}{rrcll} & m \circ f & = & m \circ g & \\ \Rightarrow & m_x \circ f_x & = & m_x \circ g_x & \text{for } x = S, OP, X, P, T, A \\ \Rightarrow & f_x & = & g_x & m_x \text{ monomorphism for } x = S, OP, X, P, T, A \\ \Rightarrow & f & = & g & \end{array}$$

Part 2 ($\Leftarrow$). Let $m = (m_{SP}, m_P, m_T, m_A) : AN_2 \to AN_3$ be a monomorphism in **AHLNet**.
In the following, the fact that m_P, m_T, m_S, m_{OP} and m_X are monomorphisms in category **Sets**, i.e. injective functions, is shown. Note again that m_A is an isomorphism.

Suppose that m_P is not injective. This proof is analogous to the proof without data types (see proof of Fact A.2).

Suppose that m_T is not injective. This proof is analogous to the proof without data types (see proof of Fact A.2).

Suppose that (m_S, m_{OP}) is not injective, i.e. (m_S, m_{OP}) is no monomorphism in category **Spec**. Construct $AN_1 = (SP_1, P_1, T_1, pre_1, post_1, cond_1, type_1, A_1)$ with $SP_1 = (S_1, OP_1, E_1, X_1)$ with

- S_1, OP_1, E_1 as described in the proof of Fact B.2.
- $X_{1_s} = \emptyset$
- $P_1, T_1 = \emptyset$
- $pre_1, post_1, cond_1, type_1 = \emptyset$
- $A_1 = V_{f_{SP}}(A_2)$

and inclusion $f : AN_1 \to AN_2$ and $g = (g_{SP}, g_P, g_T, g_A) : AN_1 \to AN_2$ and $g_{SP} = (g_S, g_{OP}, g_X) : SP_1 \to SP_2$ with

- g_{SP} as described in the proof of Fact B.2 with the empty function $g_X = f_X = \emptyset$
- $g_P, g_T = \emptyset$
- inclusion f_A
- $g_A = f_A$ if m_S is injective, otherwise $g_{A_{s_2}} = m_{s_2'}^{-1} \circ m_{s_2}$ with s_2 and s_2' as defined in the proof of Fact B.2.

Well-definedness of AN_1 and f is evident. Well-definedness of g:

- Well-definedness of g_{SP} is already proven in the proof of Fact B.2.
- $g_X : X_1 \to X_2$ is obviously well-defined since $X_1 = \emptyset$ and $g_X = \emptyset$ is the empty function.
- $g_A : A_1 \to A_2$ is well-defined since f_A is well-defined and algebra homomorphisms are restriced to isomorphisms.

Obviously, $m \circ f = m \circ g$ holds, although $f \neq g$. This is a contradiction to the fact that m is a monomorphism.

Note that m_X is (componentwise) injective by definition. □

Fact B.4 (Epimorphisms in **Sig**)**.** Surjective signature morphisms are epimorphisms in category **Sig**.

Proof. Given a signature morphism $e = (e_S, e_{OP}) : (S_1, OP_1) \to (S_2, OP_2)$.
Part 1 ($\Rightarrow$). Let e be a surjective signature morphism, i.e. e_S and e_{OP} are surjective (i.e. epimorphisms in **Sets**, as proven in Fact A.3) and $f, g : (S_2, OP_2) \to (S_3, OP_3)$ be signature morphisms with $f \circ e = g \circ e$.

$$\begin{array}{rrcll} & f \circ e & = & g \circ e & \\ \Rightarrow & f_x \circ e_x & = & g_x \circ e_x & \text{for } x = S, OP \\ \Rightarrow & f_x & = & g_x & e_x \text{ epimorphism for } x = S, OP \\ \Rightarrow & f & = & g & \end{array}$$

Part 2 ($\Leftarrow$). Let e be an epimorphism in **Sig**.
In the following, the fact that e_S and e_{OP} are epimorphisms in category **Sets**, i.e. surjective functions, is shown.

Suppose that e_S is not surjective. Then $\exists s_2 \in S_2 : \nexists s_1 \in S_1 : e_S(s_1) = s_2$. Construct (S_3, OP_3) as a copy of (S_2, OP_2) except for

- an additional sort $s_2' \in S_3$
- additional operations for this sort $op\{\{s_2 \leftarrow s_2'\}\} \in \widetilde{OP}_{3_{s_2'}} \Leftrightarrow op \in \widetilde{OP}_{2_{s_2}}$, where $\widetilde{OP}_{i_s}$ is the set of all operations containing s as one of the sorts in its signature of the specification SP_i and $\{\{s_2 \leftarrow s_2'\}\}$ is the substitution of s_2 by s_2'

Informally, (S_3, OP_3) is a copy of (S_2, OP_2) with a complete copy of sort s_2 to s_2' with all operations containing s_2 as one of the sorts in their signature. Note that the corresponding operations have the same names ("overloaded operations"). Obviously, (S_3, OP_3) is well-defined.
Next, construct inclusion $f : (S_2, OP_2) \to (S_3, OP_3)$ and $g = (g_S, g_{OP}) : (S_2, OP_s) \to (S_3, OP_3)$ with $g = f$ except for

- $g_S(s_2) = s_2'$
- $g_{OP}(op)$ maps to the overloaded operation with the same name as $f_{OP}(op)$ but a different signature

Well-definedness of the morphisms is evident. Obviously, $f \circ e = g \circ e$ holds, although $f \neq g$. This is a contradiction to the fact that e is an epimorphism.

Suppose that e_{OP} is not surjective. Then $\exists op_2 \in OP_2 : \nexists op_1 \in OP_1 : e_{OP}(op_1) = op_2$. If the signature of op_2 contains sorts without a preimage in S_1, then e_S is not surjective. Remember that the surjectivity of e_S is a necessary condition for e being an epimorphism, as already proven above. So this case can be neglected.
Construct (S_3, OP_3) as a copy of (S_2, OP_2) with an additional operation op_2' with the same signature as op_2. Obviously, (S_3, OP_3) is well-defined.
Next, construct inclusion $f : (S_2, OP_2) \to (S_3, OP_3)$ and $g = (g_S, g_{OP}) : (S_2, OP_2) \to (S_3, OP_3)$ with $g = f$ except for $g_{OP}(op_2) = op_2'$. Well-definedness of the morphisms is evident. Obviously, $f \circ e = g \circ e$ holds, although $f \neq g$. This is a contradiction to the fact that e is an epimorphism. □

Fact B.5 (Epimorphisms in **Spec**)**.** Surjective specification morphisms are epimorphisms in category **Spec**.

Proof. Given a specification morphism $e = (e_S, e_{OP}) : SP_1 \to SP_2$ with $SP_i = (S_i, OP_i, E_i)$ for $i = 1, 2$.
Part 1 ($\Rightarrow$). Let e be a surjective signature morphism. This proof is analogous to the corresponding proof in category **Sig** (see Fact B.4).
Part 2 ($\Leftarrow$). Let e be an epimorphism in **Spec**.
In the following, the fact that e_S and e_{OP} are epimorphisms in category **Sets**, i.e. surjective functions, is shown.

Suppose that e_S is not surjective. Note that **Spec**-morphisms are restricted to morphisms mapping the variables bijectively. This proof is analogous to the corresponding proof in **Sig** (see Fact B.4) with the following additional constructions:

- $E_3 = E_2 \cup E_3'$, with the same equations E_3' for the new sort defined as $e\{\{v \leftarrow v'\}\} \in E_3' \Leftrightarrow e \in E_2$, where $\{\{v \leftarrow v'\}\}$ is the substitution of all variables v of the sort s_2 by the corresponding variables v' of the sort s_2'.

Informally, SP_3 is a copy of SP_2 with a complete copy of sort s_2 to s_2' with all operations and equations containing s_2 as one of the sorts in their signature. Note that the names of the corresponding operations are equal ("overloaded operations"). SP_3 is obviously well-defined.

Suppose that e_{OP} is not surjective. This proof is analogous to the corresponding proof in **Sig** (see Fact B.4) with the following additional equations

- $E_3 = E_2 \cup E_3'$, with new equations E_3' for the new operation defined as $e\{\{op_2 \leftarrow op_2'\}\} \in E_3 \Leftrightarrow e \in E_2$, where $\{\{e_2 \leftarrow e_2'\}\}$ is the substitution of operation op_2 by the new operation op_2'

Well-definedness of this construction is evident. □

Fact B.6 (Epimorphisms in **AHLNet**)**.** Surjective **AHLNet**-morphisms are epimorphisms in the category **AHLNet**.

Proof. Let $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2, 3$ be AHL nets.
Part 1 ($\Rightarrow$). Let $e : AN_1 \to AN_2$ be a surjective **AHLNet**-morphism and $f, g : AN_2 \to AN_3$ be **AHLNet**-morphisms with $f \circ e = g \circ e$.
By definition, m_P, m_T, m_S, m_{OP} and m_X are surjective functions, i.e. epimorphisms in category **Sets**, as already proven in Fact A.3. Note again that f_A is an isomorphism, i.e. f_A is a monomorphism (and an epimorphism).

$$\begin{array}{rrcll} & f \circ e & = & g \circ e & \\ \Rightarrow & f_x \circ e_x & = & g_x \circ e_x & \text{for } x = S, OP, X, P, T, A \\ \Rightarrow & f_x & = & g_x & e_x \text{ epimorphism for } x = S, OP, X, P, T, A \\ \Rightarrow & f & = & g & \end{array}$$

Part 2 ($\Leftarrow$). Let e be an epimorphism in **AHLNet**.
In the following, the fact that e_P, e_T, e_S, e_{OP} and e_X are epimorphisms in category **Sets**, i.e. surjective functions, is shown. Note again that e_A is an isomorphism.

Suppose that e_P is not surjective. This proof is analogous to the proof without data types (see proof of Fact A.4).

Suppose that e_T is not surjective. This proof is analogous to the proof without data types (see proof of Fact A.4).

Suppose that e_S is not surjective. Then $\exists s_2 \in S_2 : \nexists s_1 \in S_1 : e_S(s_1) = s_2$ and some cases can be distinguished:

- If $\exists p_2 \in P_2 : type_2(p_2) = s_2$, then $\nexists p_1 \in P_1 : e_P(p_1) = p_2$, i.e. e_P is not surjective. Note the fact that e is an epimorphism implies that e_P is surjective, as already proven above.

- If $\exists t_2 \in T_2 : pre_2(t_2)$ contains s_2 as one of the sorts in its signature, $\nexists t_1 \in T_1 : e_T(t_1) = t_2$, i.e. e_T is not surjective. Note the fact that e is an epimorphism implies that e_T is surjective, as already proven above. This is analogous for *post* and *cond*.

- s_2 is a sort not used in the net part. Construct AN_3 as a copy of AN_2 except for

  - S_3, OP_3, E_3 as described in proof of Fact B.5
  - new additional variables for the new sort $X_{3_{s_2'}} = \{x' | x \in X_{2_{s_2}}\}$
  - $A_{3_{s_2'}} = A_{2_{s_2}}$
  - $A_{3_{op}} = A_{2_{op}}$ for all $op \in \widetilde{OP}_{3_{s_2'}}$ (see above for a definition of this set)

  Obviously, AN_3 is well-defined.
  Next, construct inclusion $f : AN_2 \to AN_3$ and $g = (g_{SP}, g_P, g_T, g_A) : AN_2 \to AN_3$ with $g_{SP} = (g_S, g_{OP}, g_X) : AN_2 \to AN_3$ with $g = f$ except for

  - g_{SP} as described in the proof of Fact B.5 with $g_X(x) = x'$
  - $g_{A_{s_2'}} = f_{A_{s_2}}$

  Well-definedness of the morphisms is evident. Obviously, $f \circ e = g \circ e$ holds, although $f \neq g$. This is a contradiction to the fact that e is an epimorphism.

Suppose that e_{OP} is not surjective. Then $\exists op_2 \in OP_2 : \nexists op_1 \in OP_1 : e_{OP}(op_1) = op_2$. If the signature of op_2 contains sorts without a preimage in S_1, then e_S is not surjective. Remember that surjectivity of e_S is a necessary condition for e being an epimorphism, as already proven above. So this case can be neglected.
If the signature of op_2 only contains sorts with a preimage in S_1, the following cases can be distinguished:

- If $\exists t_2 \in T_2 : pre_2(t_2)$ contains op_2 as one of the operations in its signature, $\nexists t_1 \in T_1 : e_T(t_1) = t_2$, i.e. e_T is not surjective. Note the fact that e is an epimorphism implies that e_T is surjective, as already proven. This is analogous for *post* and *cond*.

- op_2 is an operation not used in the net part. Construct AN_3 as a copy of AN_2 except for

  - S_3, OP_3, E_3 as described in the proof of Fact B.5
  - $A_{3_{op_2'}} = A_{2_{op_2}}$

  Obviously, AN_3 is well-defined.
  Next, construct inclusion $f : AN_2 \to AN_3$ and $g = (g_{SP}, g_P, g_T, g_A) : AN_2 \to AN_3$ with $g_{SP} = (g_S, g_{OP}, g_X) : AN_2 \to AN_3$ with $g = f$ except for

  - $g_{OP}(op_2) = op_2'$

  Well-definedness of the morphisms is evident. Obviously, $f \circ e = g \circ e$ holds, although $f \neq g$. This is a contradiction to the fact that e is an epimorphism.

Suppose that e_X is not (componentwise) surjective, i.e. $\exists e_{X_s}$ with $s \in S_1$ which is not surjective. Then $\exists x_2 \in X_{2_{e_S(s)}} : \nexists x_1 \in X_{1_s} : e_{X_s}(x_1) = x_2$. Construct AN_3 as a copy of AN_2 except for

- $X_{3_{e_S(s)}} = X_{2_{e_S(s)}} \cup \{x_2'\}$ with $x_2' \notin X_{2_{e_S(s)}}$.

Obviously, AN_3 is well-defined.
Next, construct inclusion $f : AN_2 \to AN_3$ and $g = (g_{SP}, g_P, g_T, g_A) : AN_2 \to AN_3$ with $g_{SP} = (g_S, g_{OP}, g_X) : AN_2 \to AN_3$ with $g = f$ except for

- $g_{X_{e_S(s)}}(x_2) = x_2'$

Well-definedness of the morphisms is evident. Note that g_X is (componentwise) injective by construction. Obviously, $f \circ e = g \circ e$ holds, although $f \neq g$. This is a contradiction to the fact that e is an epimorphism. □

Fact B.7 (Jointly Epimorphic Morphisms in **Sig**)**.** Jointly surjective **Sig**-morphisms are jointly epimorphic morphisms in category **Sig**. Jointly surjective **Sig**-morphisms are componentwise jointly surjective for sorts and operations.

Proof. Analogous to the proof of Fact B.4. □

Fact B.8 (Jointly Epimorphic Morphisms in **Spec**)**.** Jointly surjective **Spec**-morphisms are jointly epimorphic morphisms in category **Spec**. Jointly surjective **Spec**-morphisms are componentwise jointly surjective for sorts and operations.

Proof. Analogous to the proof of Fact B.5. □

Fact B.9 (Jointly Epimorphic Morphisms in **AHLNet**)**.** Jointly surjective **AHLNet**-morphisms are jointly epimorphic morphisms in category **AHLNet**.

Proof. Analogous to the proof of Fact B.6. □

Fact B.10 (Isomorphisms in **Sig**)**.** Bijective **Sig**-morphisms are isomorphisms in the category **Sig**.

Proof. Trivial, since this statement holds in **Sets** (see Fact A.6). □

Fact B.11 (Isomorphisms in **Spec**)**.** Bijective and strict **Spec**-morphisms are isomorphisms in the category **Spec**.

Proof. Trivial. The strictness of isomorphism $i : SP_1 \to SP_2$ is required to assure the existence of i^{-1}. □

Fact B.12 (Isomorphisms in **AHLNet**)**.** Bijective and strict **AHLNet**-morphisms are isomorphisms in the category **AHLNet**.

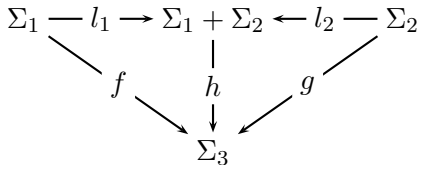
Proof. Let $i = (i_{SP}, i_P, i_T, i_A) : AN_1 \to AN_2$ with $i_{SP} = (i_S, i_{OP}, i_X) : SP_1 \to SP_2$ be an **AHLNet**-morphism for $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2$ being AHL nets.
Part 1 ($\Rightarrow$). Let i be bijective and strict.
By definition, i_x for $x = SP, P, T, X$ are bijective, i.e. isomorphisms. This statement also holds for $x = A$ because of the restriction of the morphisms to isomorphisms for the algebra component. Hence, $\exists i_x^{-1} : i_x^{-1} \circ i_x = id_{x_1} \wedge i_x \circ i_x^{-1} = id_{x_2}$ for $x = SP, P, T, A, X$.
Obviously, $i^{-1} = (i_{SP}^{-1}, i_P^{-1}, i_T^{-1}, i_A^{-1})$ is a well-defined **AHLNet**-morphism with $i^{-1} \circ i = id_{AN_1}$ and $i \circ i^{-1} = id_{AN_2}$.
Part 2 ($\Leftarrow$). Let i be an isomorphism in **AHLNet**.
Obviously, i_x for $x = SP, P, T, X$ are isomorphisms. By the restriction of **AHLNet**-morphisms, this statement also holds for $x = A$. According to Fact B.11 and Fact A.6, i_x for $x = SP, P, T, A, X$ is bijective, i.e. $i = (i_{SP}, i_P, i_T, i_A)$ is bijective.
Isomorphism i implies the existence of an inverse morphism i^{-1}. Hence, $i_{SP}^{-1}(E_2) \subseteq E_1$, i.e. i is strict. □

B.1.2 Categorical Constructions

In this subsection, binary coproducts, pushouts and pullbacks along strict injective morphisms in the category **AHLNet** are presented. Since all these constructions are defined componentwise, the same constructions are also presented in the category **Sig** and **Spec**.

Fact B.13 (Binary Coproducts in **Sig**)**.** Binary coproducts in **Sig** are componentwise coproducts for sorts and operations.

Proof. Let $\Sigma_i = (S_i, OP_i)$ for $i = 1, 2$ and $\Sigma_1 + \Sigma_2 = (S_1 + S_2, OP_1 + OP_2)$ be signatures and $l_1 = (l_{1_S}, l_{1_{OP}}) : \Sigma_1 \to \Sigma_1 + \Sigma_2$ and $l_2 = (l_{2_S}, l_{2_{OP}}) : \Sigma_2 \to \Sigma_1 + \Sigma_2$ be **Sig**-morphisms.



Part 1 ($\Rightarrow$). Let $\Sigma_1 + \Sigma_2$ be the componentwise disjoint union of Σ_1 and Σ_2 for sorts and operations and l_1, l_2 be inclusions.
Given signature $\Sigma_3 = (S_3, OP_3)$ with morphisms $f : \Sigma_1 \to \Sigma_3$ and $g : \Sigma_2 \to \Sigma_3$. Define $h = (h_S, h_{OP}) : \Sigma_1 + \Sigma_2 \to \Sigma_3$ with

- $h_S(s) = \begin{cases} f_S(s) & |s \in S_1 \\ g_S(s) & |s \in S_2 \end{cases}$
- h_{OP} analogous.

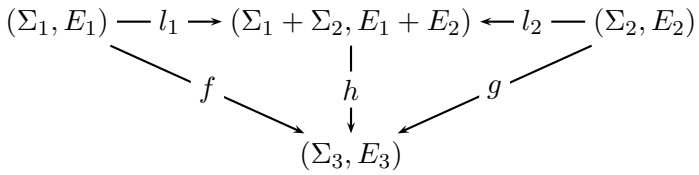
Well-definedness of h follows directly by well-definedness of f and g. By construction, $h \circ l_1 = f$ and $h \circ l_2 = g$.
Part 2 ($\Leftarrow$). Let $(\Sigma_1 + \Sigma_2, l_1, l_2)$ be the coproduct of Σ_1 and Σ_2 in **Sig**.
In the following, the facts that $\Sigma_1 + \Sigma_2$ is the componentwise disjoint union of Σ_1 and Σ_2 and l_1 and l_2 are inclusions are shown.
Assume $S_1 + S_2$ is not the disjoint union of S_1 and S_2. Several cases can be distinguished:

- $\exists s \in S_1 + S_2 \backslash (l_1(S_1) \cup l_2(S_2))$, i.e. (l_1, l_2) are not jointly surjective in **Sig**. As already proven, this is equivalent to the fact that (l_1, l_2) are not jointly epimorphic in category **Sig**. This is a contradiction to Fact D.2.
- $\exists s \in l_1(S_1) \cup l_2(S_2)$. Construct Σ_3 as componentwise disjoint union of Σ_1 and Σ_2 and inclusions $f : \Sigma_1 \to \Sigma_3$ and $g : \Sigma_2 \to \Sigma_3$. Then no unique morphism $h : \Sigma_1 + \Sigma_2 \to \Sigma_3$ with $h \circ l_1 = f$ and $h \circ l_2 = g$ exists, which is a contradiction to the fact that $\Sigma_1 + \Sigma_2$ is the coproduct in **Sig**.
- $\exists s \neq s' \in S_1 + S_2 : l_{1_S}(s) = l_{1_S}(s')$. Construct Σ_3 as componentwise disjoint union of Σ_1 and Σ_2 and $f : \Sigma_1 \to \Sigma_3$ and $g : \Sigma_2 \to \Sigma_3$ as inclusions. Obviously, no morphism $h : \Sigma_1 + \Sigma_2 \to \Sigma_3$ with $h \circ l_1 = f$ and $h \circ l_2 = g$ exists. This is a contradiction to the fact that $\Sigma_1 + \Sigma_2$ is the coproduct in **Sig**. This is analogous for l_{2_S} being non-injective.

This is analogous for $OP_1 + OP_2$. □

Fact B.14 (Binary Coproducts in **Spec**)**.** Binary coproducts in **Spec** are componentwise coproducts for sorts, operations and equations.

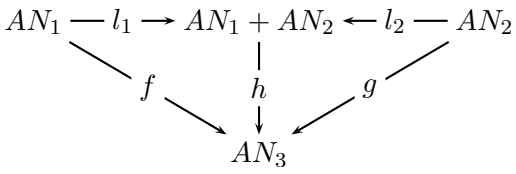
Proof.



Analogous to **Sig** with the addition that the componentwise union of equations remains to be shown.
Part 1 ($\Rightarrow$). Obviously.
Part 2 ($\Leftarrow$). Assume $E_1 + E_2$ is not the disjoint union of E_1 and E_2. Since $S_1 + S_2$ and $OP_1 + OP_2$ are the disjoint unions for the corresponding components, $\exists e \in (E_1 + E_2) \backslash (l_1(E_1) \cup l_2(E_2))$ holds by assumption. Construct $\Sigma_3 = (S_3, OP_3)$ as componentwise disjoint unions of the corresponding sets. Then a unique induced morphism $h_\Sigma : (S_1 + S_2, OP_1 + OP_2)$ with $h_\Sigma \circ l_{1_\Sigma} = f_\Sigma$ and $h_\Sigma \circ l_{2_\Sigma} = g_\Sigma$ exists. However, h_Σ is no specification morphism since $h_\Sigma^\#(e) \notin E_3$. □

Fact B.15 (Binary Coproducts in **AHLNet**)**.** Binary coproducts in **AHLNet** are componentwise coproducts for the specification $SP = (S, OP, X)$, the places P and the transitions T.

Proof. Let $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2$ and $AN_1 + AN_2 = (SP_1 + SP_2, P_1 + P_2, T_1 + T_2, pre_{1+2}, post_{1+2}, cond_{1+2}, type_{1+2}, A_1 + A_2)$ with $SP_1 + SP_2 = (S_1 + S_2, OP_1 + OP_2, E_1 + E_2, X_1 + X_2)$ be AHL nets and $l_1 = (l_{1_{SP}}, l_{1_P}, l_{1_T}, l_{1_A}) : AN_1 \to AN_1 + AN_2$ with $l_{1_{SP}} = (l_{1_S}, l_{1_{OP}}, l_{1_X})$ and $l_2 = (l_{2_{SP}}, l_{2_P}, l_{2_T}, l_{2_A}) : AN_2 \to AN_1 + AN_2$ with $l_{2_{SP}} = (l_{2_S}, l_{2_{OP}}, l_{2_X})$ be **AHLNet**-morphisms.



Part 1 ($\Rightarrow$). Let $(S_1, OP_1, E_1) + (S_2, OP_2, E_2)$, $P_1 + P_2$, $T_1 + T_2$ and $X_1 + X_2$ be coproducts and l_1, l_2 be inclusions.
Given AHL net $AN_3 = (SP_3, P_3, T_3, pre_3, post_3, cond_3, type_3, A_3)$ with $SP_3 = (S_3, OP_3, E_3, X_3)$ and **AHLNet**-morphisms $f = (f_{SP}, f_P, f_T, f_A) : AN_1 \to AN_3$ with $f_{SP} = (f_S, f_{OP}, f_X)$ and $g = (g_{SP}, g_P, g_T, g_A) : AN_2 \to AN_3$ with $g_{SP} = (g_S, g_{OP}, g_X)$. The componentwise coproducts imply that $\exists h_x : x_1 + x_2 \to x_3$ with $h_x \circ l_1 = f \wedge h_x \circ l_2 = g$ for $x = SP, P, T, X$. This statement also holds for $x = A$ because **AHLNet**-morphisms are isomorphisms on the algebra part. $h = (h_{SP}, h_P, h_T, h_A)$ is a well-defined **AHLNet**-morphism because of the well-definedness of l_i for $i = 1, 2$, f and g. By construction, $h \circ l_1 = f$ and $h \circ l_2 = g$ hold.
Part 2 ($\Leftarrow$). Let $(AN_1 + AN_2, l_1, l_2)$ be the coproduct of AN_1 and AN_2 in **AHLNet**.

In the following, the fact that $AN_1 + AN_2$ is the componentwise disjoint union of AN_1 and AN_2 and morphisms l_i for $i = 1, 2$ are inclusions, i.e. the componentwise coproduct, is shown.

Assume $P_1 + P_2$ (resp. $T_1 + T_2$) is not the coproduct of P_1 and P_2 (resp. T_1 and T_2). This is analogous to the proof without data types (see proof of Fact A.9).

Assume $(S_1, OP_1, E_1) + (S_2, OP_2, E_2)$ is not the coproduct of specifications (S_1, OP_1, E_1) and (S_2, OP_2, E_2). This is analogous to the proof of Fact B.14.

Assume $X_1 + X_2$ is not the coproduct of X_1 and X_2. Since $S_1 + S_2$ is the coproduct of S_1 and S_2, the set $(X_1 + X_2)_s$ of variables of the sort $s \in S_1 + S_2$ has a preimage in X_1 or X_2. Note that the mapping of the additional variables is restricted to injective functions. So $\exists x \in (X_1 + X_2)_s \wedge s \in S_1 : x \notin X_{1_s}$ or $\exists x \in (X_1 + X_2)_s \wedge s \in S_2 : x \notin X_{2_s}$, i.e. $l_{1_{X_s}}$ or $l_{2_{X_s}}$ is not surjective. Construct AN_3 as componentwise coproduct for the specification, the places, the transitions and the additional variables of AN_1 and AN_2. Obviously, there exist inclusions $f : AN_1 \to AN_3$ and $g : AN_2 \to AN_3$, although no morphism $h : AN_1 + AN_2 \to AN_3$ exists since x cannot be mapped injectively. This is a contradiction to the fact that $AN_1 + AN_2$ is the coproduct of AN_1 and AN_2 in **AHLNet**. □

Remark B.16 (Compatibility of Coproducts with Strict Injective AHL Net Morphisms). Coproducts in **AHLNet** are compatible with strict injective morphisms. This fact holds since coproducts in **AHLNet** are constructed componentwise in **Sets**.

Since the category **AHLNet** contains an algebra part, a technique to create the algebra part for a pushout construction in **Spec** is required. It is called amalgamation and is defined in the following part. For a more detailed introduction to amalgamation and its application area see [EM85] and [EM90].

Lemma B.17 (Amalgamation Lemma and Construction)**.** *Given pushout (1) in* **Spec** *with $SP_i = (S_i, OP_i, E_i)$ for $i = 1, 2, 3, 4$ and SP_i algebras A_i for $i = 1, 2, 3$ with $V_f(A_2) = A_1 = V_g(A_3)$. Then the amalgamated sum A_4 of A_2 and A_3 with respect to A_1, written $A_4 = A_2 +_{A_1} A_3$, is a SP_4-algebra defined by*

- $A_{4_s} =$ *if* $s \in f'_S(S_3)$ *then* $A_{3_{s_3}}$ *else* $A_{2_{s_2}}$ *with* $f'_S(s_3) = s$ *resp.* $g'_S(s_2) = s$
- op_{A_4} *analogous.*

$$\begin{array}{ccc} SP_1 & \xrightarrow{f \in \mathcal{M}} & SP_2 \\ \downarrow{\scriptstyle g} & (1) & \downarrow{\scriptstyle g'} \\ SP_3 & \xrightarrow{f' \in \mathcal{M}} & SP_4 \end{array}$$

Additionally, the following properties hold

1. *A_4 is uniquely defined by the construction*
2. *$V_{g'}(A_4) = A_2$ and $V_{f'}(A_4) = A_3$*

Vice versa, given pushout (1) as defined above and SP_4-algebra A_4, a unique representation as the amalgamated sum of $A_4 = A_2 +_{A_1} A_3$ exists, where A_1, A_2 and A_3 are defined as follows:

- $A_2 = V_{g'}(A_4)$
- $A_3 = V_{f'}(A_4)$
- $A_1 = V_f(A_2) = V_g(A_3)$

The proof of the amalgamation lemma can be found in Lemma 8.11 in [EM85].

Fact B.18 (Pushouts in **Spec**)**.** In category **Spec**, the pushout $SP_3 \xrightarrow{f'} SP_4 \xleftarrow{g'} SP_2$ of a span $SP_3 \xleftarrow{g} SP_1 \xrightarrow{f} SP_2$ for $SP_i = (S_i, OP_i, E_i)$ for $i = 1..4$ can be constructed componentwise for sorts and operations with $E_4 = g'^{\#}(E_2) \cup f'^{\#}(E_3)$.
The proof for pushouts along inclusions can be found on pages 210-212 in [EM85]. For the general case see Section 10B in [EM90], although this section contains no formal proof.

Remark B.19 (Pushouts in **Sig**). The construction of pushouts in **Sig** is analogous to the construction of pushouts in **Spec** with empty sets of equations: Pushouts in **Sig** are constructed componentwise for sorts and operations.

Fact B.20 (Pushouts in **AHLNet**)**.** In category **AHLNet**, the pushout $AN_3 \xrightarrow{f'} AN_4 \xleftarrow{g'} AN_2$ of a span $AN_3 \xleftarrow{g} AN_1 \xrightarrow{f} AN_2$ for $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1..4$ can be constructed componentwise for specifications, places, transitions and additional variables. pre_4, $post_4$, $cond_4$ and $type_4$ are induced. A_4 is given by amalgamation of A_2 and A_3 with respect to AN_1 (see Lemma B.17).
The proof of this fact is located in Lemma 5.2 in [PER95]. Note that the additional variables as well as the *type*-function are neglected in this proof.

Fact B.21 (Pullbacks in **Spec** along Strict Injective Morphisms)**.** In the category **Spec** the pullback $SP_3 \xleftarrow{g} SP_1 \xrightarrow{f} SP_2$ of a span $SP_3 \xrightarrow{f'} SP_4 \xleftarrow{g'} SP_2$ for $SP_i = (S_i, OP_i, E_i)$ for $i = 1..4$ with f' or g' being strict injective can be constructed componentwise for sorts and operations with $E_1 = f^{\#^{-1}}(E_2) \cap g^{\#^{-1}}(E_3)$ (see Fact 4.24 in [EEPT06]).

Remark B.22 (Pullbacks in **Sig** along Monomorphisms). The construction of pullbacks in **Sig** along monomorphisms is analogous to the construction of pullbacks in **Spec** with empty sets of equations: Pullbacks in **Sig** along monomorphisms are constructed componentwise for sorts and operations.

Fact B.23 (Pullbacks in **AHLNet** along Strict Injective Morphisms)**.** Pullbacks in **AHLNet** along strict injective morphisms are constructed componentwise for specifications, places, transitions and additional variables. The construction of the algebra part is obvious since algebra homomorphisms are restricted to isomorphisms.

Proof-Idea. This follows from the fact that pullbacks (along $\mathcal{M}$-morphisms) in generalized comma categories (see Definition 7.9 in [Pra07]) can be constructed componentwise if functors G_i preserve pullbacks along $\mathcal{M}_{l_i}$ morphisms (see Theorem 2 in [Pra07]). Note that the mentioned theorem does not state this fact explicitly. See Section 5 in [Pra07] for the corresponding construction of category **AHLNet**. □

B.1.3 Gluing Condition

For transforming AHL nets, a necessary and sufficient gluing condition (see Definition 2.18) for the applicability of rules to AHL nets is required. For the definition of the **AHLNet** gluing condition the **Sig** and the **Spec** gluing conditions are required. So they are defined at first.

Fact B.24 (Gluing Condition for **Sig**)**.** Let $\mathcal{M}$ be the class of all monomorphisms in **Sig**.
Given a production $p = ((S_L, OP_L) \overset{l \in \mathcal{M}}{\leftarrow} (S_K, OP_K) \overset{r \in \mathcal{M}}{\rightarrow} (S_R, OP_R))$ in **Sig** and a match $m = (m_S, m_{OP}) : (S_L, OP_L) \to (S_1, OP_1)$, the gluing points GP, the dangling points DP and the identification points IP of L are defined by

$$\begin{array}{lcl} GP(S) & = & l_S(S_K), \\ GP(OP) & = & l_{OP}(OP_K), \\ DP(S) & = & \{s \in S_L \mid \exists op \in OP_1 \backslash m_{OP}(OP_L) \text{ which contains } m_S(s) \text{ as one of} \\ & & \text{the sorts in its signature}\} \\ IP(S) & = & \{s \in S_L | \exists s' \in S_L : s \neq s' \wedge m_S(s) = m_S(s')\} \\ IP(OP) & = & \{op \in OP_L | \exists op' \in OP_L : op \neq op' \wedge m_{OP}(op) = m_{OP}(op')\} \end{array}$$

The match m satisfies the gluing condition with respect to p if and only if the following statements hold:

1. $DP(S) \cup IP(S) \subseteq GP(S)$
2. $IP(OP) \subseteq GP(OP)$

A rule in **Sig** is applicable at a match m if and only if the gluing condition is satisfied for m (see Definition 2.6 and Lemma 2.7 in [PER95] and note that the **Sig** gluing condition corresponds to the **Spec** gluing condition with the assumption that the sets of equations are empty).

Fact B.25 (Gluing Condition for **Spec**)**.** Let $\mathcal{M}$ be the class of all strict injective morphisms in **Spec**.
Given a production $p = (SP_L \overset{l \in \mathcal{M}}{\leftarrow} SP_K \overset{r \in \mathcal{M}}{\rightarrow} SP_R)$ with $SP_i = (S_i, OP_i, E_i)$ for $i = 1, L, K, R$ in **Spec** and a match $m = (m_S, m_{OP}) : SP_L \to SP_1$. Then the match m satisfies the gluing condition with respect to p if and only if

1. m satisfies the **Sig** gluing condition with respect to p (see Fact B.24)

2. the set $E_C = (E_1 \backslash m^\#(E_L)) \cup m^\#(l^\#(E_K))$ is a set of equations over the signature (S_C, OP_C), where

$$S_C = (S_1 \backslash m_S(S_L)) \cup m_S(l_S(S_K))$$
$$OP_C = (OP_1 \backslash m_{OP}(OP_L)) \cup m_{OP}(l_{OP}(OP_K))$$

A rule in **Spec** is applicable at a match m if and only if the gluing condition is satisfied for m (see Definition 2.6 and Lemma 2.7 in [PER95]).

Fact B.26 (Gluing Condition for **AHLNet**)**.** Let $\mathcal{M}$ be the class of all strict injective morphisms in **AHLNet**.
Given a production $p = (AN_L \overset{l \in \mathcal{M}}{\leftarrow} AN_K \overset{r \in \mathcal{M}}{\rightarrow} AN_R)$ with $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ and $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, L, K, R$ in **AHLNet** and a match $m = (m_{SP}, m_P, m_T, m_A) : AN_L \to AN_1$ with $m_{SP} = (m_S, m_{OP}, m_X) : SP_L \to SP_1$.

$$\begin{array}{ccccc} AN_L & \longleftarrow l \longrightarrow & AN_K & \longrightarrow r \longrightarrow & AN_R \\ \downarrow m & (1) & \downarrow f & & \\ AN_1 & \longleftarrow g \longrightarrow & AN_C & & \end{array}$$

Then the following dangling points DP are defined:

$$\begin{aligned} DP(P) = \{ \; & p \in P_L \mid \exists t \in T_1 \backslash m_T(T_L) : \\ & pre_1(t) = \sum_{i=1}^{n} \lambda_i (r_i, p_i) \text{ with } \lambda_i \neq 0 \text{ and } m_P(p) = p_i \\ & \text{for some } i \text{ or} \\ & post_1(t) = \sum_{i=1}^{n} \lambda_i (r_i, p_i) \text{ with } \lambda_i \neq 0 \text{ and } m_P(p) = p_i \\ & \text{for some } i\} \end{aligned}$$

$$\begin{aligned} DP(T_{OP}) = \{ \; & r \in T_{OP_L}(X_L) \mid \exists t \in T_1 \backslash m_T(T_L) : \\ & pre_1(t) = \sum_{i=1}^{n} \lambda_i (r_i, p_i) \text{ with } \lambda_i \neq 0 \text{ and } m^\#_{SP}(r) = r_i \\ & \text{for some } i \text{ or} \\ & post_1(t) = \sum_{i=1}^{n} \lambda_i (r_i, p_i) \text{ with } \lambda_i \neq 0 \text{ and } m^\#_{SP}(r) = r_i \\ & \text{for some } i\} \end{aligned}$$

$$\begin{aligned} DP(EQNS) = \{ \; & e \in EQNS(\Sigma_L) \\ & \mid \exists t \in T_L \backslash m_T(T_L) : m^\#_{SP}(e) \in cond_1(t)\} \end{aligned}$$

where $\Sigma_L = (S_L, OP_L)$ is the signature of AN_L

$$DP(S) = \{ \quad s \in S_L \mid \exists p \in P_1 \backslash m_P(P_L) : type_1(p) = m_S(s)\}$$

Note that the dangling points of the additional variables X are implicitly defined by $DP(T_{OP})$ and $DP(EQNS)$.
The match m satisfies the gluing condition with respect to p if and only if

1. the identification condition for the places and transitions is satisfied (see Fact A.21)
2. the **Spec** gluing condition is satisfied (see Fact B.25)
3. $DP(P) \subseteq l_P(P_K)$
4. $DP(T_{OP}) \subseteq l^{\#}_{SP}(T_{OP_L}(X_L))$
5. $DP(S) \subseteq l_S(S_K)$ (*)

A rule in **AHLNet** is applicable at a match m if and only if the gluing condition is satisfied for m.

Proof. Lemma 5.5 and Lemma 5.6 in [PER95] contain the proofs for the necessity and the sufficiency of the satisfaction of the gluing condition for the existence of a pushout complement. However, in [PER95] AHL nets are defined without a *type* function and the additional variables are neglected. Note that the identification condition for the additional variables is always satisfied since they cannot be identified. The dangling points for the additional variables are implicitly defined by $DP(T_{OP})$ and $DP(EQNS)$. So the fact that (*) is a necessary and sufficient additional condition for the existence of the context net remains to be shown.
Let $AN_C = (SP_C, P_C, T_C, pre_C, post_C, cond_C, type_C, A_C)$ with $SP_C = (S_C, OP_C, E_C, X_C)$ be an AHL net and $f : AN_K \to AN_C$ and $g : AN_C \to AN_1$ be **AHLNet**-morphisms.
Part 1 ($\Rightarrow$). Let AN_C with f and g be the pushout complement for $AN_K \xrightarrow{l} AN_L \xrightarrow{m} AN_1$. From Lemma 5.6 in [PER95] follows that the gluing condition without (*) is satisfied for the given morphisms. Assume (*) does not hold. Then $\exists s_L \in S_L, \exists p_1 \in P_1 : (\nexists p_L \in P_L : m_P(p_L) = p_1 \wedge type_1(p_1) = m_S(s_L))$ and $\nexists s_K \in S_K : l_S(s_K) = s_L$. Pushout properties imply that $\exists p_C \in P_C : g_P(p_C) = p_1$. $type_C(p_C) = s_C$ with $s_C \in S_C : g_S(s_C) = m_S(s_L)$ follows directly and, by pushout properties, $\exists s_K \in S_K : l_S(s_K) = s_L \wedge f_S(s_K) = s_C$, which is a contradiction to the fact that (*) is not satisfied.
Part 2 ($\Leftarrow$). Let the gluing condition with (*) be satisfied for $AN_K \xrightarrow{l} AN_L \xrightarrow{m} AN_1$. Then the pushout complement can be constructed as described in Lemma 5.5 in [PER95]. Note that the definitions of g' and f' in this fact are mixed up. Additionally, $type_C : P_C \to S_C$ is the restriction of $type_1$ to P_C. This function is well-defined because of (*). The additional variables X_C can be constructed as pushout complement in **Sets** for every sort. Well-definedness of f and g holds since $f = m \circ l$ and g is an inclusion. □

B.2 The Category of AHL Systems

In this section, the most important concepts of the category **AHLSystems**, with respect to this thesis, are introduced.

B.2.1 Special Morphisms

Fact B.27 (Special Morphisms in **AHLSystems**)**.** Monomorphisms (resp. epimorphisms, jointly epimorphic morphisms) in **AHLSystems** are **AHLSystems**-morphisms which are monomorphisms (resp. epimorphisms, jointly epimorphic) in **AHLNet**. Isomorphisms in **AHLSystems** are marking strict **AHLSystems**-morphisms which are isomorphisms in **AHLNet**.
The proofs therefore are analogous to the corresponding proofs in **AHLNet** (see Facts B.3 for monomorphisms, B.6 for epimorphisms, B.9 for jointly epimorphic morphisms and B.12 for isomorphisms).

B.2.2 Categorical Constructions

Binary coproducts, pushouts and pullbacks in the category **AHLSystems** are introduced in this subsection.

Fact B.28 (Binary Coproducts in **AHLSystems**)**.** The definition of binary coproducts in **AHLSystems** is the same as in **AHLNet** with the additional assumption that the injections i_1 and i_2 are marking strict.

Proof. Analogous to the proof for **AHLNet**. Obviously, the marking strictness of the injections is required to ensure the existence of the unique morphism for the universal property. □

Remark B.29 (Compatibility of Coproducts with Strict AHL Morphisms)*.* Coproducts in **AHLSystems** are compatible with strict AHL morphisms (see Definition 4.16). This fact holds since coproducts in **AHLNet** are constructed componentwise in **Sets** and coproduct inclusions are marking strict.

Fact B.30 (Pushouts in **AHLSystems** along Strict Morphisms)**.** The pushout $(AN_2, M_2) \overset{n}{\to} (AN_3, M_3) \overset{g}{\leftarrow} (AN_1, M_1)$ over the morphisms $m : (AN_0, M_0) \to (AN_1, M_1)$ and $f : (AN_0, M_0) \to (AN_2, M_2)$, where m is a strict AHL morphism (see Definition 4.16) and $AN_i = (SP_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ with $SP_i = (S_i, OP_i, E_i, X_i)$ for $i = 1, 2$, can be constructed as pushout in **AHLNet**. The marking M_3 is defined by

(1) $\forall (a_1, p_1) \in (A_1 \otimes (P_1 \backslash m_P(P_0))) : M_3(g_A(a_1), g_P(p_1)) = M_1(a_1, p_1)$

(2) $\forall (a_2, p_2) \in (A_2 \otimes (P_2 \backslash f_P(P_0))) : M_3(n_A(a_2), n_P(p_2)) = M_2(p_2)$

(3) $\forall (a_0, p_0) \in (A_0 \otimes P_0) : M_3(n_A \circ f_A(a_0), n_P \circ f_P(p_0)) = M_2(f_A(a_0), f_P(p_0))$

n also is a strict AHL morphism.

Proof. The proof of this fact is analogous to the proof of the corresponding fact without data types (see Theorem 1 in [EEH+07]). □

Fact B.31 (Pullbacks in **AHLSystems** along Strict Morphisms)**.** Given AHL morphisms $g : (AS_1, M_1) \to (AN_3, M_3)$ and $n : (AN_2, M_2) \to (AN_3, M_3)$, where n is strict (see Definition 4.16), the pullback $(AN_2, M_2) \overset{f}{\leftarrow} (AN_0, M_0) \overset{m}{\to} (AN_1, M_1)$ in **AHLSystems** is constructed as pullback in **AHLNet** and the marking M_0 is defined by

$$\forall (a_0, p_0) \in (A_0 \otimes P_0) : M_0(a_0, p_0) = M_1(m_A(a_0), m_P(p_0))$$

for P_0 being the places and A_0 being the algebra of AN_0.
m is also a strict AHL morphism.

Proof. The proof of this fact is analogous to the proof of the corresponding fact without data types (see Theorem 2 in [EEH+07]). □

B.2.3 Gluing Condition

This subsection contains the expansion of the gluing condition for AHL nets to AHL systems.

Definition B.32 (Gluing Condition for **AHLSystems**)**.** Let $\mathcal{M}$ be the class of strict AHL morphisms (see Definition 4.16) and $AS_x = (AN_x, M_x)$ with $AN_x = (SP_x, P_x, T_x, pre_x, post_x, cond_x, type_x, A_x)$ with $SP_x = (S_x, OP_x, E_x, X_x)$ for $x = 1, L, K, R$ be AHL systems.
Given a production $p = (AS_L \overset{l \in \mathcal{M}}{\leftarrow} AS_K \overset{r \in \mathcal{M}}{\to} AS_R)$ in **AHLSystems** and a match $m : AS_L \to AS_1$. The match m satisfies the gluing condition with respect to p if and only if the following statements hold:

1. The **AHLNet** gluing condition is satisfied for m with respect to p.
2. m is marking strict on places to be deleted, i.e.

$$\forall (a, p) \in (A_L \otimes (P_L \backslash l_P(P_K))) : M_L(a, p) = M_1(m_A(a), m_P(p))$$

A rule in **AHLSystems** is applicable at a match m if and only if the gluing condition is satisfied for m.
The proof of this fact is analogous to the proof of the corresponding fact without data types in [EEH+07] Section 5 since the AHL net gluing condition is a necessary and sufficient condition for the existence of the context net in category **AHLNet** (see Fact B.26).

Appendix C

Labeled P/T Systems

C.1 The Category of Labeled P/T Systems

The most important concepts of the category **PTSys**(**L**), with respect to this thesis, are introduced in this section and their correctness is proven.
In the following, let V : **PTSys**(**L**) $\to$ **PTSys** be the forgetful functor given in Definition 5.4 and $\mathcal{M}$ be the class of strict L-labeled P/T morphisms. Note that V obviously preserves $\mathcal{M}$-morphisms.

C.1.1 Special Morphisms

Fact C.1 (Special Morphisms in **PTSys**(**L**))**.** Monomorphisms (resp. epimorphisms, isomorphisms) in category **PTSys**(**L**) are monomorphisms (resp. epimorphisms, isomorphisms) in category **PTSys** (see Fact A.16).

Proof. Remember that **PTSys**(**L**) is isomorphic to the comma category **C** presented in the proof of Fact 5.3.
Part 1 ($\Leftarrow$). This follows directly by Fact A.43 (1.) in [EEPT06].
Part 2 ($\Rightarrow$). This is trivial for isomorphisms since functors preserve isomorphisms (see Definition 5.4). Let $m : (PS_B, \lambda_B) \to (PS_C, \lambda_C)$ (resp. $e : (PS_A, \lambda_A) \to (PS_B, \lambda_B)$) be a monomorphism (resp. an epimorphism) in category **PTSys**(**L**). Assume m (resp. e) is no monomorphism (resp. epimorphism) in category **PTSys**.

$$(PS_A, \lambda_A) \overset{f}{\underset{g}{\rightrightarrows}} (PS_B, \lambda_B) \xrightarrow{m} (PS_C, \lambda_C)$$

$$(PS_A, \lambda_A) \xrightarrow{e} (PS_B, \lambda_B) \overset{f}{\underset{g}{\rightrightarrows}} (PS_C, \lambda_C)$$

Then a P/T system $PS_A = (PN_A, M_A)$ (resp. $PS_B = (PN_B, M_B)$) with a labeling function λ_A (resp. λ_B) uniquely determined by f and g and two L-labeled P/T morphisms $f \neq g : (PS_A, \lambda_A) \to (PS_B, \lambda_B)$ (resp. $f \neq g : (PS_B, \lambda_B) \to (PS_C, \lambda_C)$)

can be constructed as described in part 2 of the corresponding proof in **PTNet** (see Fact A.2, resp. Fact A.4). The marking M_A (resp. M_C) is arbitrary, although it has to be ensured that f and g are well-defined. Hence, m (resp. e) is not a monomorphism (resp. epimorphism) in **PTSys**(**L**), which is a direct contradiction to the assumption that m (resp. e) is a monomorphism (resp. epimorphism) in category **PTSys**(**L**). □

Fact C.2 (Jointly Epimorphic Morphisms in **PTSys**(**L**))**.** Jointly epimorphic morphisms in category **PTSys**(**L**) are jointly epimorphic morphisms in category **PTSys**, i.e. jointly surjective **PTSys** morphisms (see Fact A.16).

Proof. Given L-labeled P/T morphisms $e_i : (PS_i, \lambda_i) \to (PS_B, \lambda_B)$ in category **PTSys**(**L**), where $PS_i = (PN_i, M_i)$ are P/T systems and $i = 1, 2$.

$$(PS_1, \lambda_1) \xrightarrow{e_1} (PS_B, \lambda_B) \underset{g}{\overset{f}{\rightrightarrows}} (PS_C, \lambda_C) \quad (PS_2, \lambda_2) \xrightarrow{e_2} (PS_B, \lambda_B)$$

Part 1 ($\Leftarrow$). Let (e_1, e_2) be jointly epimorphic in category **PTSys**. Given arbitrary **PTSys**(**L**)-morphisms $f, g : (PS_B, \lambda_B) \to (PS_C, \lambda_C)$ with $g \circ e_i = f \circ e_i$ for $i = 1, 2$. Then $g \circ e_i = f \circ e_i$ for $i = 1, 2$ holds in **PTSys**. Since (e_1, e_2) are jointly epimorphic in **PTSys**, $g = h$ holds in **PTSys**. Hence, $g = h$ also holds in **PTSys**(**L**).
Remember Fact A.16 implies that (e_1, e_2) are (componentwise) jointly surjective.
Part 2 ($\Rightarrow$). Let (e_1, e_2) be jointly epimorphic in category **PTSys**(**L**). Suppose (e_1, e_2) are not jointly epimorphic in category **PTSys**. Then a L-labeled P/T system $PS_C = (PN_C, M_C, \lambda_C)$ and morphisms $g \neq f : (PS_B, \lambda_B) \to (PS_C, \lambda_C)$ can be constructed as described in the second part of proof of Fact A.4, where M_C is an arbitrary marking that ensures the well-definedness of morphisms f and g and λ_C is uniquely determined by f and g. So (e_1, e_2) are not jointly epimorphic in **PTSys**(**L**). This is a contradiction to the assumption that (e_1, e_2) are jointly epimorphic in category **PTSys**(**L**). □

C.1.2 Categorical Constructions

Fact C.3 (Binary Coproducts in **PTSys**(**L**))**.** Binary Coproducts in **PTSys**(**L**) are binary coproducts in **PTSys**.

Proof. Given L-labeled P/T systems (PS_i, λ_i) with $PS_i = (P_i, T_i, pre_i, post_i, M_i)$ and $\lambda_i : P_i \to L$ for $i = 1, 2$.

$$\begin{array}{ccccc} (PS_1, \lambda_1) & \xrightarrow{l_1} & (PS_1 + PS_2, \lambda_1 + \lambda_2) & \xleftarrow{l_2} & (PS_2, \lambda_2) \\ & \searrow^{f_1} & \downarrow x & \swarrow_{f_2} & \\ & & (PS_3, \lambda_3) & & \end{array}$$

Part 1 ($\Leftarrow$). Let $(PS_1 + PS_2, l_1, l_2)$ be the coproduct of PS_1 and PS_2 in **PTSys** (see Fact A.17). Labeling function $\lambda_1 + \lambda_2$ is the binary coproduct in **Sets** of morphisms λ_1 and λ_2. Well-definedness of the inclusions l_1 and l_2 is evident. Given L-labeled P/T system (PS_3, λ_3) and morphisms $f_1 : (PS_1, \lambda_1) \to (PS_3, \lambda_3)$ and $f_2 : (PS_2, \lambda_2) \to (PS_3, \lambda_3)$. Then the induced morphism $x : PS_1 + PS_2 \to PS_3$ in **PTSys** is a well-defined **PTSys**(**L**)-morphism:

$$\forall p \in l_{i_P}(P_i) : \lambda_1 + \lambda_2(p) = \lambda_i(p) = \lambda_3(f_{i_P}(p)) = \lambda_3(x_P(l_{i_P}(p))) = \lambda_3(x_P(p))$$

for $i = 1, 2$. Note that l_1 and l_2 are inclusions and jointly surjective by construction.
Part 2 ($\Rightarrow$). Given coproduct $((PS_1 + PS_2, \lambda_1 + \lambda_2), l_1, l_2)$ of (PS_1, λ_1) and (PS_2, λ_2) in **PTSys**(**L**). Suppose $(PS_1 + PS_2, l_1, l_2)$ is not the binary coproduct of PS_1 and PS_2 in **PTSys**. Analogous to part 2 of the proof without labels (see Fact A.9) the same cases can be distinguished, where the labeling λ_3 of PN_3 is uniquely induced by f_1 and f_2. Additionally, there are two cases left:

1. $\exists p_1 \in P_1 : M_1(p_1) < M_1 + M_2(l_{1_P}(p_1))$. Choose the componentwise coproduct in **PTSys** with the induced labeling and two strict inclusions $((PN_3, \lambda_3), f_1 : PS_1 \to PS_3, f_2 : PS_2 \to PS_3)$ as object of comparison. Obviously, no induced L-labeled P/T morphism $x : (PS_1 + PS_2, \lambda_1 + \lambda_2) \to (PS_3, \lambda_3)$ with $x \circ l_1 = f_1$ and $x \circ l_2 = f_2$ exists. This is a contradiction to the fact that $((PS_1 + PS_2, \lambda_1 + \lambda_2), l_1, l_2)$ is the coproduct in **PTSys**(**L**).

2. $\exists p_2 \in P_2 : M_2(p_2) < M_1 + M_2(l_{2_P}(p_2))$. Analogous to the previous case.

□

Remark C.4 (Compatibility of Coproducts with Strict L-Labeled P/T Morphisms). Coproducts in **PTSys**(**L**) are compatible with strict L-labeled morphisms. This fact holds since coproducts in **PTSys**(**L**) are constructed as coproducts in **PTSys**. See also Theorem 3 in [PEL07].

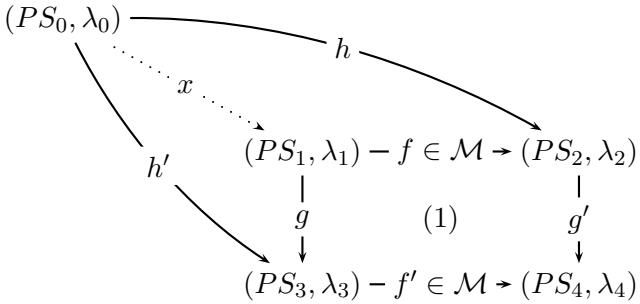
Fact C.5 (Pushouts in **PTSys**(**L**) along $\mathcal{M}$-Morphisms)**.** Pushouts in category **PTSys**(**L**) along $\mathcal{M}$-morphisms are pushouts in category **PTSys** along $\mathcal{M}$-morphisms.

Proof. Remember that **PTSys**(**L**) is isomorphic to the comma category **C** presented in the proof of Fact 5.3.
This follows directly from the properties of the comma category (see Fact 2 in [PEL07]) since functor F presented in the proof of Fact 5.3 preserves pushouts along $\mathcal{M}_1$-morphisms. □

Fact C.6 (Pullbacks in **PTSys**(**L**) along $\mathcal{M}$-Morphisms)**.** Pullbacks in category **PTSys**(**L**) are pullbacks in category **PTSys**.

Proof. Remember that **PTSys**(**L**) is isomorphic to the comma category **C** presented in the proof of Fact 5.3.

Part 1 ($\Leftarrow$). This follows directly from Fact A.43 (2.) in [EEPT06].
Part 2 ($\Rightarrow$). This follows from the comma category construction as well. Nevertheless, it is proven explicitly in this thesis since Fact A.43 (2.) in [EEPT06] does not state this conclusion explicitly.
Consider pullback (1) in **PTSys**(**L**), where $PS_i = (P_i, T_i, pre_i, post_i, M_i)$ are P/T systems and $\lambda_i : P_i \to L$ are the labeling functions of the places for $i = 0..4$, in the following diagram.



Commutativity of (1) in **PTSys** follows from functor properties and commutativity of (1) in **PTSys**(**L**):

$$V(g') \circ V(f) = V(g' \circ f) = V(f' \circ g) = V(f') \circ V(g)$$

Suppose (1) is no pullback in **PTSys**. According to the assumption and the facts that $f, f' \in \mathcal{M}$ (are injective and strict) and (1) commutes in **PTSys**, only two cases can be distinguished:

1. $\exists p_2 \in P_2, \exists p_3 \in P_3 : g'_P(p_2) = g'_P(p_3) \wedge (\nexists p_1 \in P_1 : f_P(p_1) = p_2 \wedge g_P(p_1) = p_2)$. Construct (PS_0, h, h') as pullback in **PTSys** and λ_0 as induced labeling function. Obviously, labeled P/T morphism $x : (PS_0, \lambda_0) \to (PS_1, \lambda_1)$ with $f \circ x = h$ and $g \circ x = h'$ does not exist. This is a contradiction to the fact that (1) is a pullback in **PTSys**(**L**).

2. $\exists t_2 \in T_2, \exists t_3 \in T_3 : g'_T(t_2) = g'_T(t_3) \wedge (\nexists t_1 \in T_1 : f_T(t_1) = t_2 \wedge g_T(t_1) = t_2)$. Analogous to the previous case.

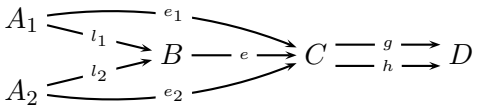
□

Appendix D

Additional Proofs

This chapter contains several proofs which are required in this thesis.

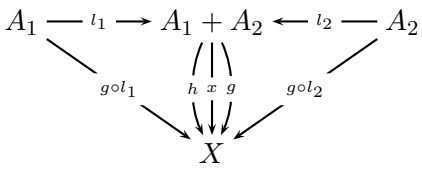
Fact D.1 (Composition of Jointly Epimorphic and Epimorphic Morphisms are Jointly Epimorphic)**.** Given a category $\mathbf{C}$, jointly epimorphic morphisms $(l_1, l_2) : A_i \to B$ for $i = 1, 2$ and epimorphism $e : B \to C$. Then it holds that $(e_1 = e \circ l_1, e_2 = e \circ l_2)$ are jointly epimorphic.



Proof. For all objects D and morphisms $g, h : C \to D$ with $g \circ e_i = h \circ e_i$ for $i = 1, 2$ the following holds:
$g \circ e_i = h \circ e_i \Leftrightarrow g \circ e \circ l_i = h \circ e \circ l_i$. Since (l_1, l_2) are jointly epimorphic and e is an epimorphism, $g = h$ holds. □

Fact D.2 (Coproduct Morphisms are Jointly Epimorphic)**.** Let $\mathbf{C}$ be a category with binary coproducts. Then the following statement holds:
If $(A_1 + A_2, l_1, l_2)$ is the binary coproduct of A_1 and A_2 then (l_1, l_2) are jointly epimorphic (see Definition A.16 in [EEPT06]).

Proof.



Let $g, h : A_1 + A_2 \to X$ be morphisms with $g \circ l_i = h \circ l_i$ for $i = 1, 2$. Then there exist morphisms $g \circ l_1 : A_1 \to X$ and $g \circ l_2 : A_2 \to X$. The universal property of the coproduct construction implies the existence of a unique morphism $x : A_1 + A_2 \to X$

with $x \circ l_i = g \circ l_i$ for $i = 1, 2$. The uniqueness of x implies that $g = x$. Since $h \circ l_i = g \circ l_i$ for $i = 1, 2$, the uniqueness of x leads to $h = x$ as well. So it is proven that $g = h$. This implies that (l_1, l_2) are jointly epimorphic. □

Fact D.3 (Pushouts in **Sets** along Monomorphisms)**.** Given the following commutative diagram (1) in **Sets**, where all of the given morphisms are monomorphisms, then (1) is a pushout if and only if the following statements hold:

1. $\forall d \in g'(B) \cap f'(C) : \exists a \in A : f(a) = b \wedge g(a) = c$ with $g'(b) = d \wedge f'(c) = d$
2. (g', f') are jointly surjective

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow{\scriptstyle g} & (1) & \downarrow{\scriptstyle g'} \\ C & \xrightarrow{f'} & D \end{array}$$

Proof. **Part 1** ($\Rightarrow$). Let (1) be a pushout in **Sets**.

1. Let $d \in g'(B) \cap f'(C)$. Suppose that $\nexists a \in A : f(a) = b \wedge g(a) = c$ with $g'(b) = d \wedge f'(c) = d$. Let $D' = D \backslash \{d\} \cup \{b, c\}$ and $\tilde{f}' : C \to D'$ and $\tilde{g}' : B \to D'$ with
$$\tilde{f}'(x) = \begin{cases} f'(x) & |x \neq c \\ c & |x = c \end{cases}$$
$$\tilde{g}'(x) = \begin{cases} g'(x) & |x \neq b \\ b & |x = b \end{cases}$$
Then no induced morphism $x : D \to D'$ with $x \circ g' = \tilde{g}'$ and $x \circ f' = \tilde{f}'$ exists, which is a contradiction to the assumption that (1) is a pushout in **Sets**.
2. Obviously, since (1) is a pushout, (g', f') are jointly surjective.

Part 2 ($\Leftarrow$). Let the following statements hold: $\forall d \in g'(B) \cap f'(C) : \exists a \in A : f(a) = b \wedge g(a) = c$ with $g'(b) = d \wedge f'(c) = d$ and (g', f') are jointly surjective. Since (1) commutes, only the universal property remains to be shown. Given object X and morphisms $h : B \to X$ and $k : C \to X$. The induced morphism $x : D \to X$ is defined for all $d \in D$ by
$$x(d) = \begin{cases} h(b) & |d = g'(b) \in g'(B) \\ k(c) & |d = f'(c) \in f'(C) \backslash g'(B) \end{cases}$$

1. Well-definedness of x. Obviously, since h and k are well-defined and (g', f') are jointly surjective.
2. $x \circ g' = h$ follows directly from definition of x.

3. $x \circ f' = k$. For $d \in f'(C) \backslash g'(B)$ it follows directly. Let $d \in g'(B) \cap f'(C)$, i.e. $\exists b \in B : g'(b) = d \wedge \exists c \in C : f'(c) = d$. According to the assumption, $\exists a \in A : g'(f(a)) = f'(g(a))$ and $x(d) = h(b) = k(c)$, i.e. $k(d) = x(d) = x(f'(c))$, follow. Note that d always has at least one preimage in B or C because of the jointly surjectivity of (g', f').

4. Uniqueness of x with respect to the commutativity. Suppose that $x' : D \to X$ with $x' \circ g' = h$ and $x' \circ f' = k$ exists. Obviously, $\forall b \in B : \exists d \in D : g'(b) = d$ and it holds that $x'(d) = h(b) = x(d)$. Analogously, $\forall c \in C : \exists d \in D : f'(c) = d$ and it holds that $x'(d) = k(c) = x(d)$. Since (g', f') are jointly surjective, it is shown that $x = x'$.

□

Bibliography

[AGG08] AGG Homepage, April 2008. Attributed Graph Grammar System, `http://tfs.cs.tu-berlin.de/agg`.

[AHS90] Jiří Adámek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories*. Wiley-Interscience, New York, NY, USA, 1990.

[BTS00] Paolo Bottoni, Gabriele Taentzer, and Andy Schürr. Efficient Parsing of Visual Languages based on Critical Pair Analysis and Contextual Layered Graph Transformation. In *VL '00: Proceedings of the 2000 IEEE International Symposium on Visual Languages (VL'00)*, page 59, Washington, DC, USA, 2000. IEEE Computer Society.

[ECL08] Eclipse Homepage, April 2008. `http://www.eclipse.org`.

[EEH+07] H. Ehrig, C. Ermel, K. Hoffmann, J. Padberg, and U. Prange. Concurrency in Reconfigurable Place/Transition Systems: Independence of Net Transformations as Well as Net Transformations and Token Firing. Technical Report 2007/02, Technische Universität Berlin, Institut für Softwaretechnik und Theoretische Informatik, 2007.

[EEPT06] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, and Gabriele Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs. Springer, 2006.

[EGdL+05] Karsten Ehrig, Esther Guerra, Juan de Lara, Laszló Lengyel, Tihamér Levendovszky, Ulrike Prange, Gabriele Taentzer, Dániel Varró, and Szilvia V. Gyapay. Model Transformation by Graph Transformation: A Comparative Study. In *MTiP '05: Proceedings of the International Workshop on Model Transformations in Practice*, 2005.

[EH86] Hartmut Ehrig and Annegret Habel. Graph Grammars with Application Conditions. In G. Rozenberg and A. Salomaa, editors, *The Book of L*, pages 87–100. Springer, 1986.

[EHKP91] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and Concurrency in High-Level Replacement Systems. *Mathematical Structures in Computer Science*, 1:361–404, 1991.

[EHKPP91] Hartmut Ehrig, Annegret Habel, Hans-Jörg Kreowski, and Francesco Parisi-Presicce. From Graph Grammars to High Level Replacement Systems. In *Proceedings of the 4th International Workshop on Graph-Grammars and Their Application to Computer Science*, volume 532 of *Lecture Notes in Computer Science*, pages 269–291, London, UK, 1991. Springer.

[EKMR99] H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation: Vol. 3: Concurrency, Parallelism, and Distribution*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1999.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1985.

[EM90] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2: Module Specifications and Constraints*, volume 21 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1990.

[EMC+01] Hartmut Ehrig, Bernd Mahr, F. Cornelius, M. Große-Rhode, and P. Zeitz. *Mathematisch-strukturelle Grundlagen der Informatik*. Springer, second edition, 2001.

[EP04] H. Ehrig and J. Padberg. Graph Grammars and Petri Net Transformations. In *Lectures on Concurrency and Petri Nets Special Issue Advanced Course PNT*, volume 3098, pages 496–536. Springer, 2004.

[EPPH06] Hartmut Ehrig, Julia Padberg, Ulrike Prange, and Annegret Habel. Adhesive High-Level Replacement Systems: A New Categorical Framework for Graph Transformation. *Fundam. Inf.*, 74(1):1–29, 2006.

[Erm96] C. Ermel. Anforderungsanalyse eines medizinischen Informationssystems mit Algebraischen High-Level-Netzen. Technical Report 96-15, Technische Universität Berlin, Institut für Softwaretechnik und Theoretische Informatik, 1996. (Masters Thesis TU Berlin).

[for08] forMAlNET Homepage, April 2008. `http://tfs.cs.tu-berlin.de/formalnet`.

[HHT96] A. Habel, R. Heckel, and G. Taentzer. Graph Grammars with Negative Application Conditions. *Special issue of Fundamenta Informaticae*, 26(3,4):287–313, 1996.

[HHT02] Jan Hendrik Hausmann, Reiko Heckel, and Gabi Taentzer. Detection of Conflicting Functional Requirements in a Use Case-Driven Approach: A Static Analysis Technique based on Graph Transformation. In *ICSE*

'02: Proceedings of the 24th International Conference on Software Engineering, pages 105–115, New York, NY, USA, 2002. ACM.

[HME05] K. Hoffmann, T. Mossakowski, and H. Ehrig. High-Level Nets with Nets and Rules as Tokens. In *Proc. of 26th Intern. Conf. on Application and Theory of Petri Nets and other Models of Concurrency*, volume 3536 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2005.

[Hof02] K. Hoffmann. Algebraic Higher Order Nets: Graphs and Petri Nets as Tokens. In M. Wirsing, D. Pattinson, and R. Henicker, editors, *Proceedings of the 16th International Workshop on Algebraic Development Techniques (WADT 2002)*, pages 87–88. LMU Munich, 2002.

[HW95] Reiko Heckel and Annika Wagner. Ensuring Consistency of Conditional Graph Rewriting - A Constructive Approach. *Electronic Notes in Theoretical Computer Science*, 2, 1995.

[KKS97] Hans-Jörg Kreowski, Sabine Kuske, and Andy Schürr. Nested Graph Transformation Units. *International Journal on Software Engineering and Knowledge Engineering*, 7(4):479–502, 1997.

[KMPP05] Manuel Koch, L. V. Mancini, and Francesco Parisi-Presicce. Graph-based Specification of Access Control Policies. *J. Comput. Syst. Sci.*, 71(1):1–33, 2005.

[Lam07] Leen Lambers. Adhesive High-Level Replacement Systems with Negative Application Conditions. Technical report, Technische Universität Berlin, Institut für Softwaretechnik und Theoretische Informatik, 2007.

[LEO06] L. Lambers, H. Ehrig, and F. Orejas. Conflict Detection for Graph Transformation with Negative Application Conditions. In *Proc. Third International Conference on Graph Transformation (ICGT'06)*, volume 4178, pages 61–76, Natal, Brazil, September 2006. Springer.

[LEOP08] L. Lambers, H. Ehrig, F. Orejas, and U. Prange. Parallelism and Concurrency in Adhesive High-Level Replacement Systems with Negative Application Conditions. In H. Ehrig, J. Pfalzgraf, and U. Prange, editors, *Proceedings of the ACCAT workshop at ETAPS 2007*, Electronic Notes in Theoretical Computer Science. Elsevier, 2008. to appear.

[LO04] M. Llorens and J. Oliver. Structural and Dynamic Changes in Concurrent Systems: Reconfigurable Petri Nets. *IEEE Transactions on Computers*, 53(9):1147–1158, 2004.

[LO06a] M. Llorens and J. Oliver. A Basic Tool for the Modeling of Marked-Controlled Reconfigurable Petri Nets. *ECEASST*, 2:13, 2006.

[LO06b] M. Llorens and J. Oliver. Marked-Controlled Reconfigurable Workflow Nets. In *SYNASC*, pages 407–413. IEEE Computer Society, 2006.

[LS04] S. Lack and P. Sobociński. Adhesive Categories. In *Foundations of Software Science and Computation Structures, FoSSaCS '04*, volume 2987 of *Lecture Notes in Computer Science*, pages 273–288. Springer, 2004.

[MM90] José Meseguer and Ugo Montanari. Petri Nets Are Monoids. *Information and Computation*, 88(2):105–155, 1990.

[MTR05] Tom Mens, Gabriele Taentzer, and Olga Runge. Detecting Structural Refactoring Conflicts using Critical Pair Analysis. *Electronic Notes in Theoretical Computer Science*, 127(3):113–128, April 2005.

[NRT92] M. Nielsen, G. Rozenberg, and P. S. Thiagarajan. Elementary Transition Systems. *Theoretical Computer Science 96*, 1992.

[PEL07] Ulrike Prange, Hartmut Ehrig, and Leen Lambers. Construction and Properties of Adhesive and Weak Adhesive High-Level Replacement Categories. *Applied Categorical Structures*, 2007.

[PER95] Julia Padberg, Hartmut Ehrig, and Leila Ribeiro. Algebraic High-Level Net Transformation Systems. *Mathematical Structures in Computer Science*, 5(2):217–256, 1995.

[Pra07] U. Prange. Algebraic High-Level Nets as Weak Adhesive HLR Categories. *Electronic Communications of the EASST*, 2:1–13, 2007.

[Pra08] U. Prange. Towards Algebraic High-Level Systems as Weak Adhesive HLR Categories. In H. Ehrig, J. Pfalzgraf, and U. Prange, editors, *Proceedings of the ACCAT workshop at ETAPS 2007*, Electronic Notes in Theoretical Computer Science. Elsevier, 2008. to appear.

[PU03] Julia Padberg and Milan Urbásek. Rule-Based Refinement of Petri Nets: A Survey. In *Petri Net Technology for Communication-Based Systems*, pages 161–196, 2003.

[Rei85] W. Reisig. *Petri Nets, An Introduction*, volume 4 of *EATCS, Monographs on Theoretical Computer Science*. Springer, 1985.

[RON08] Ron-Editor Homepage, April 2008. `http://tfs.cs.tu-berlin.de/roneditor`.

[Roz97] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1997.

[RPL+08] Alexander Rein, Ulrike Prange, Leen Lambers, Kathrin Hoffmann, and Julia Padberg. Negative Application Conditions for Reconfigurable Place/Transition Systems. *Electronic Communications of the EASST*, 2008. to appear.

[Ull08] Conny Ullrich. Reconfigurable Open AHL Systems. Technical report, Technische Universität Berlin, Institut für Softwaretechnik und Theoretische Informatik, 2008. Diploma Thesis, to appear.