Technische Universität Berlin

Forschungsberichte
der Fakultät IV – Elektrotechnik und Informatik

Modelling Evolution of Communication Platforms and Scenarios based on Transformations of High-Level Nets and Processes

(Extended Version)

Karsten Gabriel

Bericht-Nr. 2011 – 08
ISSN 1436-9915

Modelling Evolution of Communication Platforms and Scenarios based on Transformations of High-Level Nets and Processes

Extended Version

Karsten Gabriel

Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany

kgabriel@cs.tu-berlin.de

Bericht-Nr. 2011/08
ISSN 1436-9915

Modelling Evolution of Communication Platforms and Scenarios based on Transformations of High-Level Nets and Processes

Extended Version

Karsten Gabriel

kgabriel@cs.tu-berlin.de
Integrated Graduate Program IGP H-C3, Technische Universität Berlin, Germany

Abstract

Algebraic High-Level (AHL) nets are a well-known modelling technique based on Petri nets with algebraic data types, which allows to model the communication structure and the data flow within one modelling framework. Transformations of AHL-nets – inspired by the theory of graph transformations – allow in addition to modify the communication structure. Moreover, high-level processes of AHL-nets capture the concurrent semantics of AHL-nets in an adequate way. Altogether we obtain a powerful integrated formal specification technique to model and analyse all kinds of communication based systems, especially different kinds of communication platforms.

In this paper we show how to model the evolution of communication platforms and scenarios based on transformations of Algebraic High-Level Nets and Processes. All constructions and results are illustrated by a running example showing the evolution of Apache Wave platforms and scenarios. The evolution of platforms is modelled by the transformation of AHL-nets and that of scenarios by the transformation of AHL-net processes. The first main result shows under which conditions AHL-net processes can be extended if the corresponding AHL-net is transformed. This result can be applied to show the extension of scenarios for a given platform evolution. The second main result shows how AHL-net processes can be transformed based on a special kind of transformation for AHL-nets, corresponding to action evolution of platforms. Finally, we briefly discuss the case of multiple action evolutions.

1 Introduction

High-level nets based on low-level Petri nets [Pet62, Roz87, Rei85] and data types in ML have been studied as coloured Petri nets by Jensen [Jen91] and – using algebraic data types – as algebraic high-level (AHL) nets in [Rei90, PER95, ER97].

Inspired by the theory of graph transformations [Ehr79, Roz97] transformations of AHL-nets were first studied in [PER95] which – in addition to the token game – also allow to modify the net structure by rule based transformations.

The concept of processes in Petri nets is essential to model not only sequential, but especially concurrent firing behaviour. A process of a low-level Petri net N is given by an occurrence net K together with a net morphism $p : K \to N$. Processes of high-level nets AN are often defined as processes $p : K \to Flat(AN)$ of the corresponding low-level net $Flat(AN)$, called flattening of AN. However, this is not really adequate, because the flattening is in general an infinite net and the data type structure is lost. For this reason high-level processes for algebraic high-level nets have been introduced in [EHP+02, Ehr05], which are high-level net morphisms $p : K \to AN$ based on a suitable concept of high-level occurrence nets K.

The main aim of this paper is to give a comprehensive introduction to the integrated framework of transformations of algebraic high-level nets and processes and to show how this can be applied to modern communication platforms.

In previous papers it was shown already how to use this framework to model communication platforms like Skype [HM10] and Google Wave [EG11]. In this paper we show how our integrated framework can

be used to model basic aspects of Apache Wave [Apa11b]. In Section 2 we introduce a small example of an Apache Wave platform, which is also used as running example for the following sections.

In Section 3 we introduce AHL-nets together with high-level processes in the sense of [Ehr05]. Rule based transformations in analogy to graph transformation systems [Roz97] are introduced in Section 4 for AHL-nets and AHL-processes and applied to the evolution of Apache Wave communication platforms and waves. The first main result presented in Section 5 shows under which condition an AHL-process can be extended if the corresponding AHL-net is transformed. This result can be applied to show the extension of scenarios for a given platform evolution. The second main result presented in Section 6 shows how AHL-net processes can be transformed based on a special kind of transformation for AHL-nets, corresponding to action evolution of platforms. Moreover, we briefly discuss the case of multiple action evolutions.

Finally, the conclusion in Section 7 includes a summary of the paper.

2 Evolution of Apache Wave Platforms and Scenarios

In this section we introduce our main case study Apache Wave which is a communication platform that was originally developed by the company Google [Goo11] as Google Wave. Google itself has stopped the development of Google Wave, but the development is continued by the Apache Software Foundation [Apa11a] as Apache Wave [Apa11b].

One of the most interesting aspects of Apache Wave is the possibility to make changes on previous conributions. Therefore, in contrast to email, text chat or forums, due to possible changes the resulting data of the communication does not necessarily give a comprehensive overview on all interactions of the communication. For this reason, in Apache Wave for every communication there is a history allowing the users to replay interactions of the communication step by step. So for the modelling of Apache Wave it is necessary that we do not only model the systems and the communication but also the history of the communication.

We have chosen Apache Wave as running example for this paper because it includes typical modern features of many other communication systems, such as near-real-time communication. This means that different users can simultaneously edit the same document, and changes of one user can be seen almost immediately by the other users.

Note that we do not focus on the communication between servers and clients in this contribution but on the communication between users. For details on the modeling of the more technical aspects of the server-to-server and client-to-server communication we refer to [Yon10].

In Apache Wave users can communicate and collaborate via so-called waves. A wave is like a document which can contain diverse types of data that can be edited by different invited users. The changes that are made to a wave can be simultaneously recognized by the other participating users. In order to keep track of the changes that have been made, every wave contains also a history of all the actions in that wave.

Apache Wave supports different types of extensions which are divided into gadgets and robots. The extensions are programs that can be used inside of a wave. The difference between gadgets and robots is that gadgets are not able to interact with their environment while robots can be seen as automated users that can independently create, read or change waves, invite users or other robots, and so on. This allows robots for example to do real-time translation or highlighting of texts that are written by different users of a wave. Clearly, it is intended to use different robots for different tasks and it is desired that multiple robots interact without conflicts. This makes the modeling and analysis of Apache Wave very important in order to predict possible conflicts or other undesired behavior of robots.

In [EG11] we have already shown that Google Wave (and thus also Apache Wave) can be adequately modelled using algebraic high-level (AHL) nets, which is an integration of the modeling technique of low-level Petri nets [Pet62, Roz87, Rei85] and algebraic data types [EM85].

Figure 1 shows a small example of the structure of an AHL-net *Platform* which has 3 places and 3 transitions with firing conditions, where the pre and post arcs are labelled with variables of an algebraic signature. The AHL-net *Platform* models an Apache Wave platform with some basic features like the creation of new waves, modifications to existing waves, and the invitation of users to a wave which are modeled by the transitions *new wavelet*, *modify text* and *invite user*.

A wavelet is a part of a wave that contains a user ID, a list of XML documents and a set of users

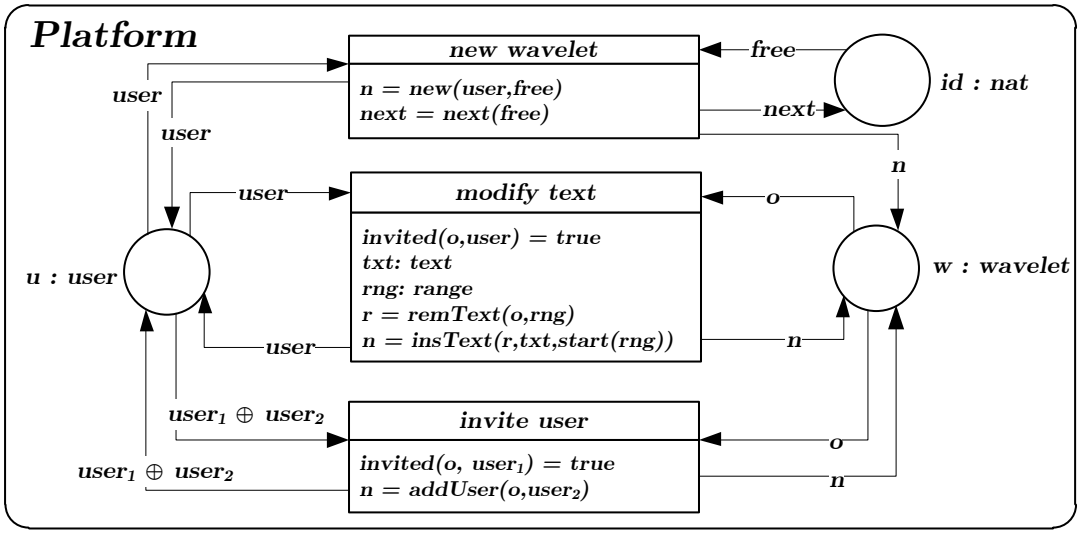


Figure 1: AHL-net *Platform* for an Apache Wave platform

which are invited to modify the wavelet. For simplicity we model in our example only the simple case that every wavelet contains only one single document and the documents contain only plain text. In order to obtain a more realistic model one has to extend the used algebraic data part of the model given by the signature Σ-$Wave$ shown in Table 1 and the Σ-$Wave$-algebra A in Table 2 in Section 3.

The transitions of the net contain firing conditions in the form of equations over the signature Σ-$Wave$. In order to fire a transition there has to be an assignment v of the variables in the environment of the transition such that the firing conditions are satisfied in the algebra A. The pair (t, v) is then called a consistent transition assignment. Moreover, there have to be suitable data elements in the pre domain of the transition. For example, in order to fire the transition *modify text* we need a wavelet on the place w and a user on the place u that can be assigned by the variables o respectively $user$ such that the user is invited to the selected wavelet. Furthermore, we need a text txt, a pair of natural numbers rng and a new wavelet n such that n is the wavelet which is obtained by deleting all text in the range specified by rng and by inserting the text txt on the starting position of rng into the original wavelet o.

The assignment v then determines a follower marking which is computed by removing the assigned data tokens in the pre domain of the transition and adding the assigned data tokens in the post domain. In the case of the transition *modify text* this means that we remove the old wavelet from the place w and replace it by a new wavelet which contains the modified text. For more details on the operational semantics of AHL-nets we refer to [Ehr05].

As we have shown in [EG11] a suitable modelling technique for waves together with their histories are AHL-processes with instantiations. Fig. 2 shows an example of an AHL-process *Wave* which abstractly models a wave that contains two wavelets created by possibly different users.

Another interesting aspect of the modelling of Apache Wave are dynamic changes to the structure of the platform. Using rule-based transformation of AHL-nets [PER95] in the sense of graph transformation [Roz97], we can delete and add features, leading to a new platform. Figure 3 shows a net *Platform′* which is an adaption of our example *Platform* where the *modify text* transition has been replaced. In the new version we have a new transition *modify and log* which for every modification to a wave creates a log entry with information about the user, the position and the text of the modification.

Since it is possible that the communication platform is modified at runtime there may already exist some waves that correspond to the old version of the platform. In some cases that correspondence could be violated by the modification of the platform.

An intuitive solution is to apply the modification of the platform also to the wave, replacing all occurrences of the old feature with the new one. This leads to a new wave model $Wave'$ depicted in Fig. 4. The three transitions have been replaced by transitions that have the same firing conditions as the transition *modify and log* and there are new *log* places in the post domain of the new transitions. In Sections 5 and 6 we present general constructions to obtain a new correspondence between an existing wave and a modified platform based on platform evolution under certain conditions.

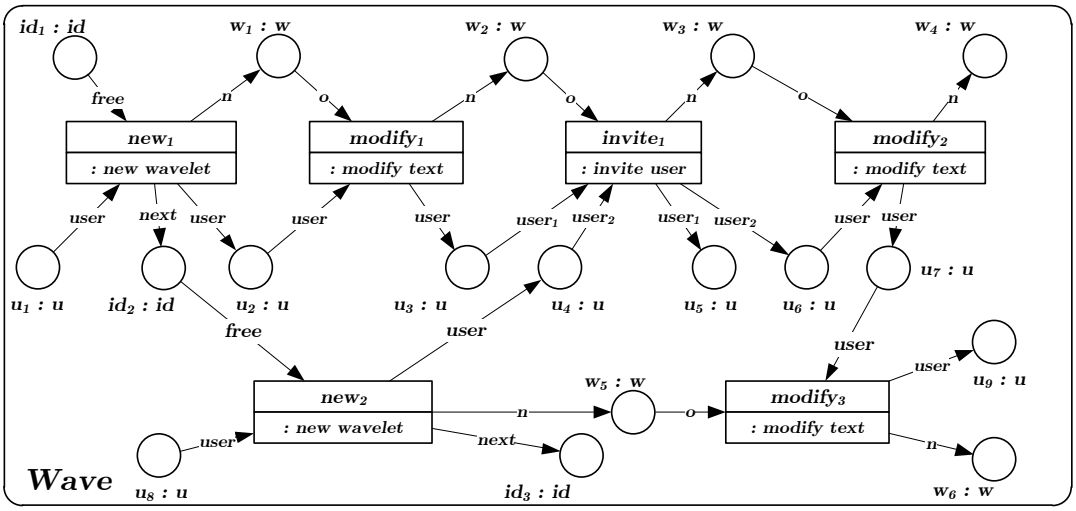


Figure 2: AHL-process $Wave$ of a wave

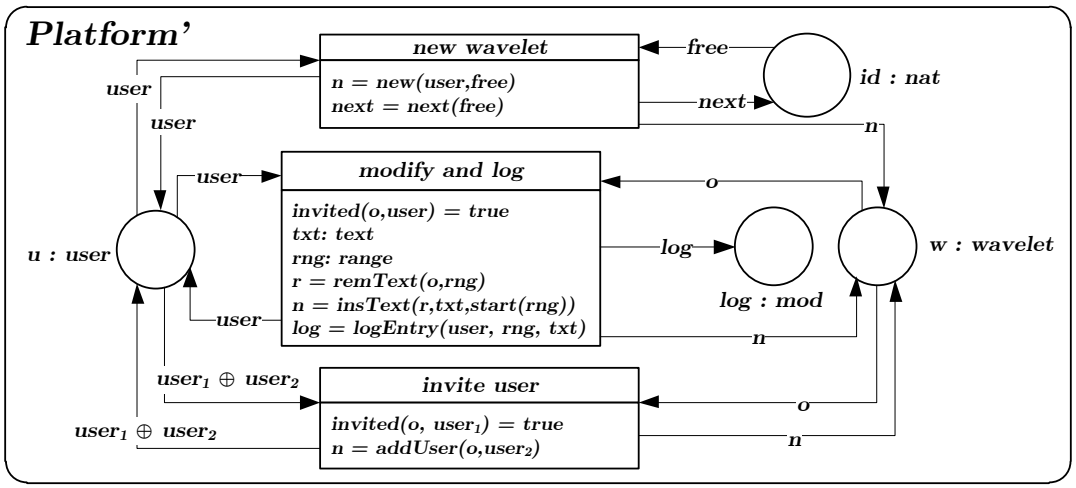


Figure 3: Modified AHL-net $Platform'$

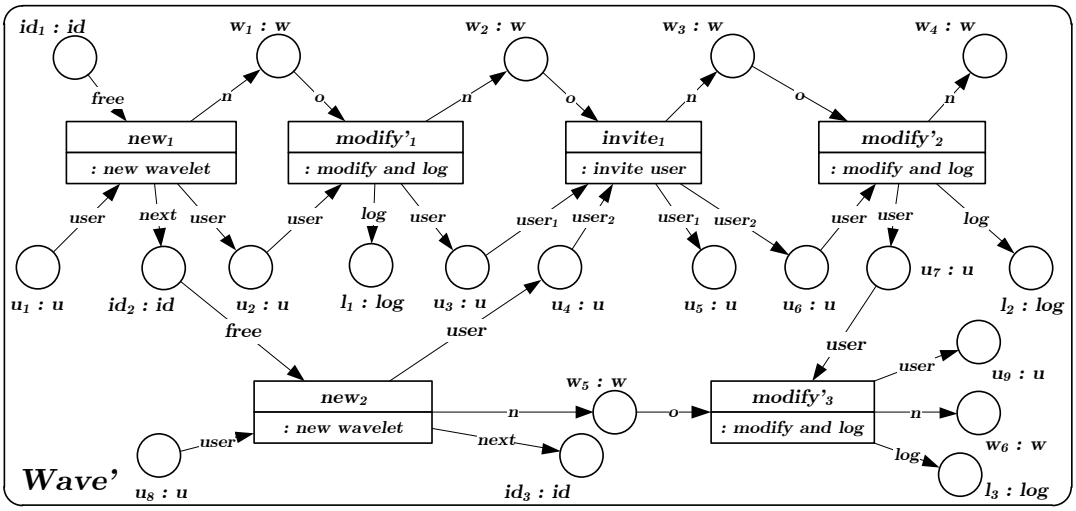


Figure 4: Modified AHL-process $Wave'$

3 Modelling of Communication Platforms and Scenarios with AHL-Nets and Processes

In the following we review the definition of AHL-nets and their processes from [Ehr05, EHP+02] based on low-level nets in the sense of [MM90], where $X^{\oplus}$ is the free commutative monoid over the set X. Note that $s \in X^{\oplus}$ is a formal sum $s = \sum_{i=1}^{n} \lambda_i x_i$ with $\lambda_i \in \mathbb{N}$ and $x_i \in X$ meaning that we have λ_i copies of x_i in s and for $s' = \sum_{i=1}^{n} \lambda'_i x_i$ we have $s \oplus s' = \sum_{i=1}^{n} (\lambda_i + \lambda'_i) x_i$.

Definition 1 (Algebraic High-Level Net)**.** An *algebraic high-level (AHL-) net*

$$AN = (\Sigma, P, T, pre, post, cond, type, A)$$

consists of

- a signature $\Sigma = (S, OP; X)$ with additional variables X;
- a set of places P and a set of transitions T;
- pre- and post domain functions $pre, post : T \to (T_\Sigma(X) \otimes P)^{\oplus}$;
- firing conditions $cond : T \to \mathcal{P}_{fin}(Eqns(\Sigma; X))$;
- a type of places $type : P \to S$ and
- a Σ-algebra A

where the signature $\Sigma = (S, OP)$ consists of sorts S and operation symbols OP, $T_\Sigma(X)$ is the set of terms with variables over X,

$$(T_\Sigma(X) \otimes P) = \{(term, p) | term \in T_\Sigma(X)_{type(p)}, p \in P\}$$

and $Eqns(\Sigma; X)$ are all equations over the signature Σ with variables X.

An AHL-net morphism $f : AN_1 \to AN_2$ 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

(1) $(id \otimes f_P)^{\oplus} \circ pre_1 = pre_2 \circ f_T$ and $(id \otimes f_P)^{\oplus} \circ post_1 = post_2 \circ f_T$,
(2) $cond_2 \circ f_T = cond_1$ and
(3) $type_2 \circ f_P = type_1$.

The category defined by AHL-nets (with signature Σ and algebra A) and AHL-net morphisms is denoted by **AHLNets** where the composition of AHL-net morphisms is defined componentwise for places and transitions.

Note that it is also possible to define a category of AHL-nets with different signatures and algebras which requires that the morphisms not only contain functions for places and transitions but also a signature morphism together with a generalized algebra morphism (for details see [PER95]).

The firing behaviour of AHL-nets is defined analogously to the firing behaviour of low-level nets. The difference is that in the high-level case all tokens are equipped with data values. Moreover, for the activation of a transition t, we additionally need an assignment asg of the variables in the environment of the transition, such that the assigned pre domain is part of the given marking and the firing conditions of the transition are satisfied. This assignment is then used to compute the follower marking, obtained by firing of transition t with assignment asg.

Definition 2 (Marking, Firing Behaviour)**.** A marking $M = \sum_{i=1}^{n} \lambda_i (a_i, p_i) \in (A \otimes P)^{\oplus}$ of an AHL-net AN means that place p_i contains $\lambda_i \in \mathbb{N}$ data tokens $a_i \in A_{type(p_i)}$.
Given an AHL-net AN with marking M a transition $t \in T$ is enabled under M and an assignment $asg : Var(t) \to A$, if all firing conditions $cond(t)$ are satisfied in A for asg and we have enough token in the pre domain of t, i.e. $pre_A(t, asg) \leq M$, where

$$pre_A(t, asg) = \sum_{i=1}^{n} (\overline{asg}(term_i), p_i) \quad \text{for} \quad pre(t) = \sum_{i=1}^{n} (term_i, p_i)$$

with $term_i \in T_{OP}(X)$ and $\overline{asg}(term_i)$ is the evaluation of $term_i$ under asg. In this case the follower marking M' is given by

$$M' = M \ominus pre_A(t, asg) \oplus post_A(t, asg).$$

Remark 1 (AHL-Nets with Individual Tokens). In contrast to the firing behaviour defined in Def. 2 it is also possible to define a marking over a set I of individuals and a marking function $m : I \to A \otimes P$ assigning each individual to a pair of a data element and a place. This makes it possible to distinguish the single tokens of a marking.

In order to fire a transition under a given marking it is then necessary to specify a token selection (M, m, N, n) where $M \subseteq I$ is the set of individuals which are consumed by the transition, N is a set of newly created individuals with $(I \setminus M) \cap N = \emptyset$ and $m : M \to A \otimes P$, $n : N \to A \otimes P$ are corresponding marking functions. If a selection together with a consistent transition assignment (t, asg) meets the *token selection condition*:

$$\sum_{i \in M} m(i) = pre_A(t, asg) \quad \text{and} \quad \sum_{i \in N} n(i) = post_A(t, asg)$$

then t is asg-enabled and the follower marking (I', m') can be computed by

$$I' = (I \setminus M) \cup N, \quad m' : I' \to A \otimes P \ \text{ with } \ m'(x) = \begin{cases} m(x), \text{ if } x \in I \setminus M; \\ n(x), \text{ if } x \in N. \end{cases}$$

Although this *individual token approach* is more complicated than the *collective token approach* in Def. 2 it has some benefits like the possibity to formulate transformation rules which can not only change the net structure but also the marking of an AHL-net. For more details we refer to [MGE+10]. In this paper we still use the collective approach but we will also research processes of AHL-nets with individual tokens in the future.

Example 1 (Apache Wave Platform)**.** The model of an Apache Wave platform in Fig. 1 is an AHL-net

$$Platform = (\Sigma\text{-}Wave, P, T, pre, post, cond, type, A)$$

where the signature Σ-$Wave$ is shown in Table 1 and the Σ-$Wave$-algebra A is shown in Table 2. This signature and algebra is also used for all the following examples.

Let us consider the marking

$$M = (Alice, u) \oplus (Bob, u) \oplus (1, id) \oplus ((0, \{Alice, Bob\}, \epsilon), w)$$

of the AHL-net *platform* in Fig. 1 which means that we have two users *Alice* and *Bob* on the place u, a free ID 1 and an empty wavelet with ID 0 on place w where *Alice* and *Bob* are invited. An assignment $asg : \{user, txt, rng, o, n\} \to A$ with $asg(user) = Alice$, $asg(txt) = Hello\ Bob$, $asg(rng) = (0, 0)$, $asg(o) = (0, \{Alice, Bob\}, \epsilon)$ and $asg(n) = (0, \{Alice, Bob\}, Hello\ Bob)$ satisfies the firing conditions of the transition *modify text*. By firing the transition *modify text* with assignment asg we obtain the follower marking

$$M' = (Alice, u) \oplus (Bob, u) \oplus, (1, id) \oplus ((0, \{Alice, Bob\}, Hello\ Bob), w)$$

where the assigned text *Hello Bob* has been inserted at position 0 into the assigned wavelet.

Remark 2 (Morphisms Preserve Firing Behaviour). Given an AHL-net morphism $f : AN_1 \to AN_2$ the firing behaviour is preserved, i.e. for $M_1' = M_1 \ominus pre_{1,A}(t, asg) \oplus post_{1,A}(t, asg)$ in AN_1 we have $M_2' = M_2 \ominus pre_{2,A}(f_T(t), asg) \oplus post_{2,A}(f_T(t), asg)$ in AN_2 with $M_2 = \sum_{i=1}^{n} (a_i, f_P(p_i))$ for $M_1 = \sum_{i=1}^{n} (a_i, p_i)$ and similar M_2' constructed from M_1'.

Now, we introduce AHL-process nets based on low-level occurrence nets (see [GR83]) and AHL-processes according to [Ehr05, EHP+02]. The net structure of a high-level occurrence net has similar properties like a low-level occurrence net, but it captures a set of different concurrent computations due to different initial markings. In fact, high-level occurrence nets can be considered to have a set of initial

Table 1: Signature Σ-$Wave$

sorts:	bool, nat, mod, range, text, user, wavelet	
opns:	true, false : $\rightarrow$ bool	next : nat $\rightarrow$ nat
	start, end : range $\rightarrow$ nat	new : user nat $\rightarrow$ wavelet
	addUser : user wavelet $\rightarrow$ wavelet	invited : wavelet user $\rightarrow$ bool
	len : text $\rightarrow$ nat	sub : text range $\rightarrow$ text
	insText : wavelet text nat $\rightarrow$ wavelet	remText : wavelet range $\rightarrow$ wavelet
	logEntry : user range text $\rightarrow$ mod	
vars:	free, next : nat	log : mod
	rng : range	txt : text
	user, user_1, user_2 : user	o, n, r : wavelet

Table 2: Σ-$Wave$-algebra A

$A_{bool} = \{T, F\}$	$A_{nat} = \mathbb{N}$
$A_{user} = \{a, \ldots, z, A, \ldots, Z\}^*$	$A_{text} = \{a, \ldots, z, A, \ldots, Z, \ldots\}^*$
$A_{wavelet} = A_{nat} \times \mathcal{P}(A_{user}) \times A_{text}$	$A_{range} = A_{nat} \times A_{nat}$
$A_{mod} = A_{user} \times A_{range} \times A_{text}$	

$true_A = T \in A_{bool}$

$\text{false}_A = F \in A_{bool}$

$\text{start}_A : A_{range} \rightarrow A_{nat}$

$(s, e) \mapsto s$

$\text{end}_A : A_{range} \rightarrow A_{nat}$

$(s, e) \mapsto e$

$next_A : A_{nat} \rightarrow A_{nat}$

$n \mapsto n + 1$

$new_A : A_{user} \times A_{nat} \rightarrow A_{wavelet}$

$(u, id) \mapsto (id, \{u\}, \epsilon)$

$addUser_A : A_{user} \times A_{wavelet} \rightarrow A_{wavelet}$

$(u, (id, uset, t) \mapsto (id, uset \cup \{u\}, t)$

$invited_A : A_{wavelet} \times A_{user} \rightarrow A_{bool}$

$$(u, (id, uset, t)) \mapsto \begin{cases} T & \text{, if } u \in uset; \\ F & \text{, else.} \end{cases}$$

$len_A : A_{text} \rightarrow A_{nat}$

$$t \mapsto \begin{cases} 0 & \text{, if } t = \epsilon; \\ 1 + len_A(t_1 \ldots t_n) & \text{, if } t = t_0 \ldots t_n. \end{cases}$$

$sub_A : A_{text} \times A_{range} \rightarrow A_{text}$

$$(t, (s, e)) \mapsto \begin{cases} \epsilon & \text{, if } e < s \text{ or } len_A(t) \leq s; \\ t_s \ldots t_n & \text{, if } t = t_0 \ldots t_m, s \leq e, s < m \text{ and } n = min(m, e). \end{cases}$$

$insText_A : A_{wavelet} \times A_{text} \times A_{nat} \rightarrow A_{wavelet}$

$((id, uset, t), nt, pos) \mapsto (id, uset, sub_A(t, (0, pos - 1)).nt.sub_A(t, (pos, len_A(t))))$

$remText_A : A_{wavelet} \times A_{range} \rightarrow A_{wavelet}$

$((id, uset, t), (s, e)) \mapsto (id, uset, sub_A(t, (0, s)).sub_A(t, (e, len_A(t))))$

$logEntry_A : A_{user} \times A_{range} \times A_{text} \rightarrow A_{mod}$

$(u, r, t) \mapsto (u, r, t)$

markings for the input places, whereas there is only one implicit initial marking of the input places for low-level occurrence nets.

Moreover, in a low-level occurrence net with an initial marking there is for any complete order of transitions compatible with the causal relation a corresponding firing sequence once there is a token on all input places. This is a consequence of the fact that in an occurrence net the cuasal relation is finitary. In the case of high-level occurrence nets an initial marking additionally contains data values and in general some of the firing conditions in a complete order of transitions are not satisfied. Hence, even in the case that the causal relation is finitary, we cannot expect to have complete firing sequences.

In order to ensure a complete firing sequence in a high-level occurrence net there has to be an "instantiation" of the occurrence net (see [Ehr05]). Instantiations, however, are not considered explicitely in this paper. In the following definition of AHL-process nets, in contrast to occurrence nets, we omit the requirement that the causal relation has to be finitary, because this is not a meaningful requirement for our application domain.

Definition 3 (Algebraic High-Level Process Net)**.** An AHL-process net K is an AHL -net

$$K = (\Sigma, P, T, pre, post, cond, type, A)$$

such that for all $t \in T$ with $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ and notation $\bullet t = \{p_1, \ldots, p_n\}$ and similarly $t\bullet$ we have

1. (*Unarity*): $\bullet t, t\bullet$ are sets rather than multisets for all $t \in T$, i.e. for $\bullet t$ the places $p_1 \ldots p_n$ are pairwise distinct. Hence $|\bullet t| = n$ and the arc from p_i to t has a unary arc-inscription $term_i$.
2. (*No Forward Conflicts*): $\bullet t \cap \bullet t' = \emptyset$ for all $t, t' \in T, t \neq t'$
3. (*No Backward Conflicts*): $t\bullet \cap t'\bullet = \emptyset$ for all $t, t' \in T, t \neq t'$
4. (*Partial Order*): the causal relation $<_K \subseteq (P \times T) \cup (T \times P)$ defined by the transitive closure of $\{(p,t) \in P \times T \mid p \in \bullet t\} \cup \{(t,p) \in T \times P \mid p \in t\bullet\}$ is a strict partial order, i. e. the partial order is irreflexive.

AHL-process nets (with signature Σ and algebra A) together with AHL-net morphisms between AHL-process nets form the full subcategory **AHLPNets** $\subseteq$ **AHLNets**.

Note that an AHL-process net with a finitary causal relation is an AHL-occurrence net as defined in [Ehr05].

We define the sets of input and output places of an AHL-process nets as the sets of places which are not in the post respectively pre domain of a transition:

Definition 4 (Input and Output Places)**.** Given an AHL-process net K. We define the set $IN(K)$ of input places of K as

$$IN(K) = \{p \in P_K \mid \nexists t \in T_K : p \in t\bullet\}$$

and similar the set $OUT(K)$ of output places of K as

$$OUT(K) = \{p \in P_K \mid \nexists t \in T_K : p \in \bullet t\}$$

Note that the properties of AHL-process nets are reflected by AHL-morphisms which is stated in the following lemma.

Lemma 1 (AHL-Morphisms Reflect AHL-Process Nets)**.** *Given an AHL-morphism $f : K_1 \to K_2$. If K_2 is an AHL-process net then also K_1.*

Proof Idea. The unarity of K_2 together with the fact that AHL-morphisms preserve pre and post conditions imply that all non-injectively matched parts have equal structures. Thus, K_1 basically has the same structural properties as K_1 which means that it does also satisfy the structural conditions to be an AHL-process net. For a detailed proof see Appendix A.1. □

An AHL-process of an AHL-net AN is defined as an AHL-morphism from an AHL-process net K into the net AN. Note that from the preservation of firing behaviour by AHL-morphisms (see Remark 2) it follows that a firing sequence in K corresponds to a firing sequence in AN. Thus, due to the conflict-free and acyclic structure of AHL-process nets, an AHL-process mp of an AHL-net AN models a part of the semantics of AN which – up to concurrency and possibly different data values – does not contain any branches or iterations.

Definition 5 (AHL-Process)**.** An AHL-process of an AHL-net AN is an AHL-net morphism $mp : K \to AN$ where K is an AHL-process net.

The category **Proc**(**AN**) of AHL-processes of an AHL-net AN is defined as the full subcategory of the slice category **AHLNets** $\setminus AN$ such that the objects are AHL-processes. This means that the objects of **Proc**(**AN**) are AHL-process morphisms $mp : K \to AN$ and the morphisms of the category are AHL-net morphisms $f : K_1 \to K_2$ such that diagram (1) below commutes.

The category **AHLProcs** of all AHL-processes is defined as full subcategory of the arrow category **AHLNets**$^{\to}$ such that the objects are AHL-processes, and the morphisms are pairs (f^*, f) of AHL-process net morphisms and AHL-morphisms such that diagram (2) below commutes.

$$\begin{array}{ccc} K_1 & \xrightarrow{f} & K_2 \\ & {\scriptstyle mp_1}\searrow \quad \swarrow{\scriptstyle mp_2} & \\ & AN & \end{array} \qquad (1) \qquad\qquad \begin{array}{ccc} K_1 & \xrightarrow{f^*} & K_2 \\ {\scriptstyle mp_1}\downarrow & & \downarrow{\scriptstyle mp_2} \\ AN_1 & \xrightarrow[f]{} & AN_2 \end{array} \qquad (2)$$

Example 2 (Scenario)**.** Figure 2 shows an AHL-process $wave : Wave \to Platform$ where the mappings of the process are indicated with colons, e. g. $u_1 : u$ means that the place u_1 in the AHL-process net *Wave* is mapped to the place u in the AHL-net *Platform* in Figure 1. The AHL-process describes an abstract scenario in the Apache Wave platform in which two wavelets are created with consecutive IDs by possibly two different users. Moreover, the creator of the first wavelet does a modification to the wavelet, and it is open if this happens before or after the creation of the second wavelet. After that the creator of the second wavelet is invited to the first one, and does modifications to the first and then to the second wavelet.

4 Evolution of Communication Platforms and Transformation of Scenario Nets

Due to the possibility to evolve the Apache Wave platforms by adding, removing or changing features we need also techniques that make it possible to evolve the corresponding model of a platform. For this reason we introduce rule-based AHL-net transformations [PER95] in the sense of graph transformations [Roz97].

A production (or transformation rule) for AHL-nets specifies a local modification of an AHL-net. It consists of a left-hand side, an interface which is the part of the left-hand side which is not deleted and a right-hand side which additionally contains newly created net parts.

Definition 6 (Productions for AHL-Nets)**.** A *production for AHL-nets* is a span $\varrho : L \xleftarrow{l} I \xrightarrow{r} R$ of injective AHL-morphisms. We call L the left-hand side, I the interface, and R the right-hand side of the production ϱ. In most examples l and r are inclusions.

$$L \xleftarrow{l} I \xrightarrow{r} R$$

Example 3 (Production for Platform Evolution)**.** Figure 5 shows a production *insertLog* for AHL-nets that can be used for the evolution of an Apache Wave platform. The production describes a local modification that removes a transition *modify text* and inserts a new transition *modify and log* and a new place *log*. Moreover, the newly created transition is connected to the former environment of the removed transition.

In order to add the new parts as specified in the right-hand side of a production to an AHL-net we define a gluing construction based on the gluing of its place and transition components in the category of **Sets** in the following sense.

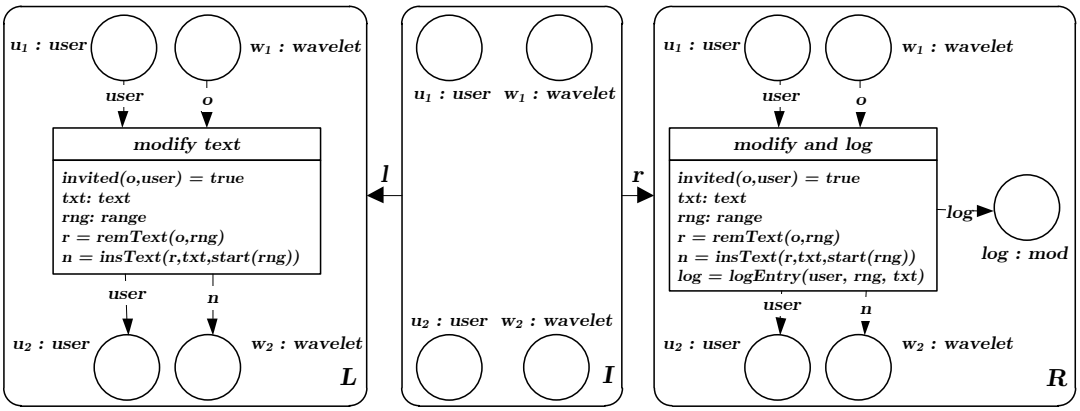


Figure 5: AHL-net production *insertLog* for the evolution of platforms

Definition 7 (Gluing of Sets)**.** Given sets A, B and C, and functions $f_1 : A \to B$, $f_2 : A \to C$. The gluing D of B and C along A (or more precisely along f_1 and f_2), written $D = B +_A C$, is defined as the quotient $D = (B \uplus C)/_\equiv$ where $\equiv$ is the smallest equivalence relation containing the relation

$$\sim \;=\; \{(f_1(a), f_2(a)) \mid a \in A\}.$$

This means that we transitively identify all those elements in $B \uplus C$ which are commonly mapped by the same interface element. Moreover, we obtain functions $g_1 : B \to D$ and $g_2 : C \to D$ with $g_1(b) = [b]_\equiv$ for all $b \in B$, and $g_2(c) = [c]_\equiv$ for all $c \in C$.

$$\begin{array}{ccc} A & \xrightarrow{f_1} & B \\ {\scriptstyle f_2}\downarrow & (\mathbf{PO}) & \downarrow{\scriptstyle g_1} \\ C & \xrightarrow[g_2]{} & D \end{array}$$

Fact 1 (Pushout of Sets)**.** *The diagram (PO) in Def. 7 is a* pushout diagram *in the category* **Sets***, i. e. (PO) commutes and has the following universal property: For all sets X and functions $h_1 : B \to X$, $h_2 : C \to X$ with $h_1 \circ f_1 = h_2 \circ f_2$ there exists a unique $h : D \to X$ with $h \circ g_1 = h_1$ and $h \circ g_2 = h_2$.*

Proof. See Fact 2.17 in [EEPT06]. □

The gluing of AHL-nets over a given interface can be defined as the component-wise gluing in **Sets**. Due to the fact that the gluing in **Sets** is also a pushout, we obtain also unique induced pre, post, condition and type functions, leading to a well-defined AHL-net as shown in [Gab10].

Definition 8 (Gluing of AHL-Nets)**.** Given two AHL-net morphisms $f_1 : AN_0 \to AN_1$ and $f_2 : AN_0 \to AN_2$ the *gluing AN_3 of AN_1 and AN_2 along f_1 and f_2*, written $AN_3 = AN_1 +_{(AN_0, f_1, f_2)} AN_2$, with $AN_x = (\Sigma, P_x, T_x, pre_x, post_x, cond_x, type_x, A)$ for $x = 0, 1, 2, 3$ is constructed as follows:

- $T_3 = T_1 +_{T_0} T_2$ with $f'_{1,T}$ and $f'_{2,T}$ as pushout (2) of $f_{1,T}$ and $f_{2,T}$ in **Sets**.
- $P_3 = P_1 +_{P_0} P_2$ with $f'_{1,P}$ and $f'_{2,P}$ as pushout (3) of $f_{1,P}$ and $f_{2,P}$ in **Sets**
- $pre_3(t) = \begin{cases} f'^{\oplus}_{1,P} \circ pre_1(t_1) & \text{, if } f'_{1,T}(t_1) = t; \\ f'^{\oplus}_{2,P} \circ pre_2(t_2) & \text{, if } f'_{2,T}(t_2) = t. \end{cases}$
- $post_3(t) = \begin{cases} f'^{\oplus}_{1,P} \circ post_1(t_1) & \text{, if } f'_{1,T}(t_1) = t; \\ f'^{\oplus}_{2,P} \circ post_2(t_2) & \text{, if } f'_{2,T}(t_2) = t. \end{cases}$
- $cond_3(t) = \begin{cases} cond_1(t_1) & \text{, if } f'_{1,T}(t_1) = t; \\ cond_2(t_2) & \text{, if } f'_{2,T}(t_2) = t. \end{cases}$

- $type_3(p) = \begin{cases} type_1(p_1) & \text{, if } f'_{1,P}(p_1) = p; \\ type_2(p_2) & \text{, if } f'_{2,P}(p_2) = p. \end{cases}$

- $f'_1 = (f'_{1,P}, f'_{1,T})$ and $f'_2 = (f'_{2,P}, f'_{2,T})$.

$$\begin{array}{ccc} AN_0 & \xrightarrow{f_1} & AN_1 \\ {\scriptstyle f_2}\downarrow & (1) & \downarrow{\scriptstyle f'_1} \\ AN_2 & \xrightarrow[f'_2]{} & AN_3 \end{array} \qquad \begin{array}{ccc} T_0 & \xrightarrow{f_{1,T}} & T_1 \\ {\scriptstyle f_{2,T}}\downarrow & (2) & \downarrow{\scriptstyle f'_{1,T}} \\ T_2 & \xrightarrow[f'_{2,T}]{} & T_3 \end{array} \qquad \begin{array}{ccc} P_0 & \xrightarrow{f_{1,P}} & P_1 \\ {\scriptstyle f_{2,P}}\downarrow & (3) & \downarrow{\scriptstyle f'_{1,P}} \\ P_2 & \xrightarrow[f'_{2,P}]{} & P_3 \end{array}$$

Fact 2 (Pushout of AHL-Nets)**.** *The diagram (1) in Def. 8 is a* pushout diagram *in the category* **AHLNets***, i. e. (1) commutes and it has the following universal property: For all AHL-nets AN'_3 and AHL-morphisms $h_1 : AN_1 \to AN'_3$, $h_2 : AN_2 \to AN'_3$ with $h_1 \circ f_1 = h_2 \circ f_2$ there exists a unique AHL-morphism $h : AN_3 \to AN'_3$ such that $h \circ f'_1 = h_1$ and $h \circ f'_2 = h_2$.*

Proof-Idea. The pushouts (2) and (3) provide unique functions $h_P : P_3 \to P'_3$, $h_T : T_3 \to T'_3$ which together form an AHL-morphism $h = (h_P, h_T) : AN_3 \to AN'_3$ satisfying the universal property. A detailed proof can be found in [Gab10]. □

Example 4 (Evolution of Apache Wave Platform)**.** With the gluing of AHL-nets we can use the production *insertLog* in Fig. 5 to describe an evolution of Apache Wave platforms. Figure 6a shows a gluing of the two AHL-nets L and $Platform_0$ over the interface I leading to our example AHL-net $Platform$ from Section 2. Note that the morphism k maps the places in I non-injectively to the corresponding places u and w. As a result the gluing does not only glue the transition *modify text* to the places that are mapped by the same interface places but also the equally mapped places from *Platform*$_0$ are glued together in the net *Platform*. On the other hand, the AHL-nets L and *Platform*$_0$ can be considered as a decomposition of the AHL-net *Platform*.

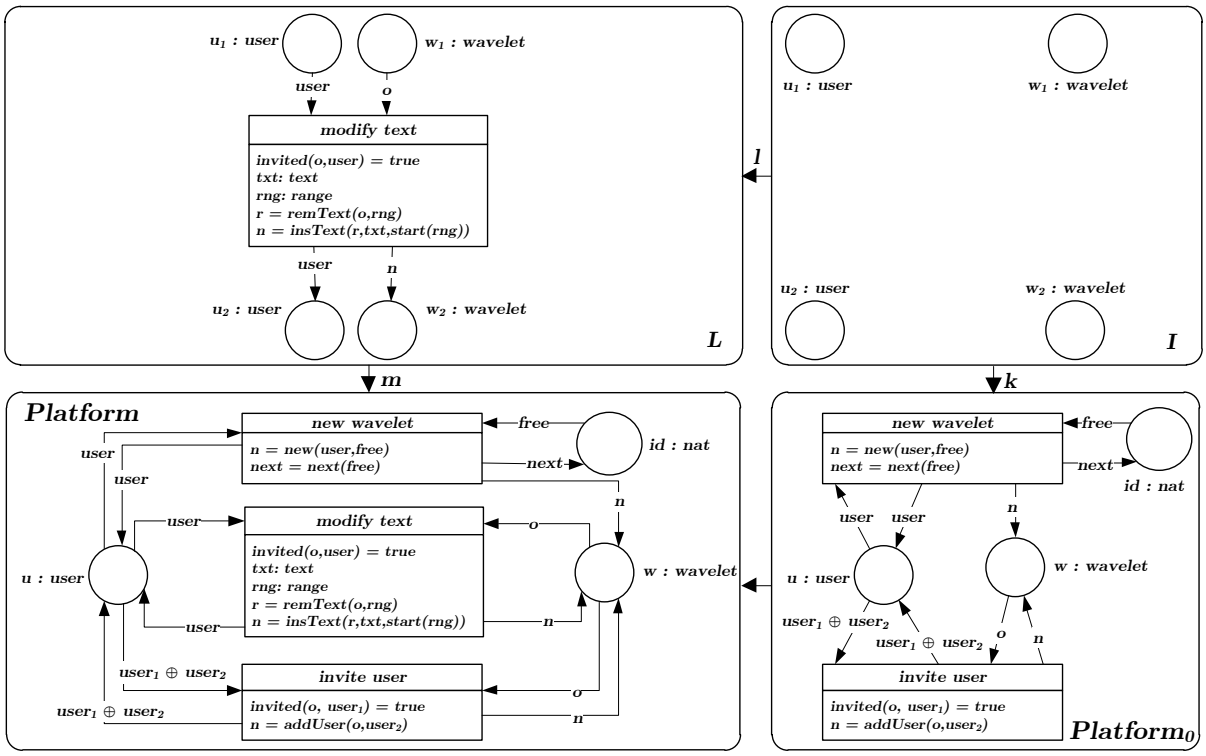
Consider now the right-hand side of the production *insertLog* in Fig. 5. Using the morphisms $r : I \to R$ and $k : I \to Platform_0$ we obtain the AHL-net *Platform*$'$ in Fig. 3 as gluing of R and *Platform*$_0$ over the interface I as shown in Fig. 6b.

The combination of both gluings describes a direct transformation of AHL-nets *Platform* $\Rightarrow$ *Platform*$'$ in the sense of Def. 9 below. The transformation uses the production *insertLog* at match $m : L \to$ *Platform*, replacing the transition *modify text* by *modify and log*.

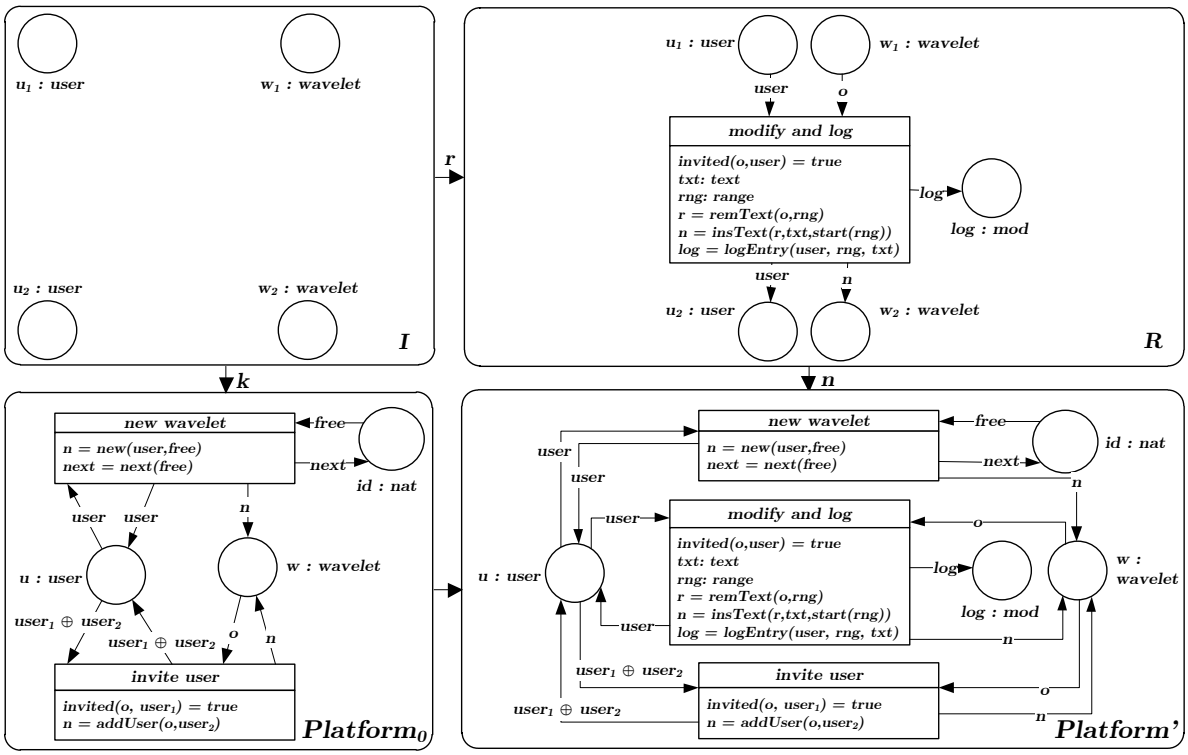
Definition 9 (Direct Transformation of AHL-Nets)**.** Given a production $\varrho : L \xleftarrow{l} I \xrightarrow{r} R$ and a (match) morphism $m : L \to AN$ in **AHLNets**.
Then a *direct transformation* $AN \overset{(p,m)}{\Rightarrow} AN'$ in **AHLNets** is given by pushouts (1) and (2) in **AHLNets**.
A transformation of AHL-nets is a sequence $AN_0 \overset{(p_1,m_1)}{\Rightarrow} AN_1 \cdots \overset{(p_n,m_n)}{\Rightarrow} AN_n$ of direct transformations, written $AN_0 \Rightarrow^* AN_n$.

$$\begin{array}{ccccc} L & \xleftarrow{l} & I & \xrightarrow{r} & R \\ {\scriptstyle m}\downarrow & (1) & {\scriptstyle c}\downarrow & (2) & {\scriptstyle n}\downarrow \\ AN & \xleftarrow{d} & C & \xrightarrow{e} & AN' \end{array}$$

Remark 3 (Modelling of Token-Game with Transformation)*.* For AHL-nets with individual tokens (see Remark 1) there is a similar definition for the rule-based direct transformation of AHL-nets with individual tokens (see [MGE+10]). It allows an alternative way to model the firing behaviour of AHL-nets by rule-based transformation. For every consistent transition assignment (t, asg) (see Def. 2) of an AHL-net with individual tokens ANI enabled under a token selection $S = (M, m, N, n)$ (see Remark 1) there is a corresponding *transition rule* $\varrho(t, S, asg)$ such that there is an equivalence between the firing of (t, asg) via S and the *canonical direct transformation* of ANI using the rule $\varrho(t, S, asg)$. For more details we refer to [MGE+10].



(a) Gluing of AHL-nets L and $Platform_0$



(b) Gluing of AHL-nets R and $Platform_0$

Figure 6: Direct transformation of AHL-nets $Platform \Rightarrow Platform'$

The following gluing condition is a necessary and sufficient condition for the existence of a direct transformation of AHL-nets. In order to satisfy the gluing condition by a production ϱ under a match m some of the places and transitions in the AHL-net AN in the codomain of m must not be deleted by application of the production. The preimages of these elements in the left-hand side of the production are called identification points and dangling points.

The identification points are the preimages of places and transitions which are mapped non-injectively by the match m. The dangling points are the preimages of places which occur in the pre or post conditions of a transition which is matched, and therefore cannot be deleted by application of the production.

Definition 10 (Gluing Condition for AHL-Nets)**.** Given a production $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ for AHL-nets and an AHL-morphism $m : L \rightarrow AN$. We define the set of identification points[1]

$$\begin{aligned} IP \quad = \quad & \{x \in P_L \mid \exists x' \neq x : m_P(x) = m_P(x')\} \cup \\ & \{x \in T_L \mid \exists x' \neq x : m_T(x) = m_T(x')\} \end{aligned}$$

the set of dangling points[2]

$$\begin{aligned} DP = \{p \in P_L \quad | \quad & \exists t \in T_{AN} \setminus m_T(T_L), term \in T_\Sigma(X)_{type(p)} : \\ & (term, m_P(p)) \leq pre_{AN}(t) \oplus post_{AN}(t)\} \end{aligned}$$

and the set of gluing points[3]

$$GP = l_P(P_I) \cup l_T(T_I)$$

We say that ϱ and m satisfy the gluing condition if $IP \cup DP \subseteq GP$.

$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & I & \xrightarrow{\ r\ } & R \\ {\scriptstyle m}\downarrow & & & & \\ AN & & & & \end{array}$$

Fact 3 (Direct Transformation of AHL-Nets)**.** *Given a production for AHL-nets $\varrho = (L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R)$ and a match $m : L \rightarrow AN$. The production ϱ is applicable on match m, i. e. there exists a context AHL-net AN_0 in the diagram below, such that (1) is pushout, iff ϱ and m satisfy the gluing condition in* **AHLNets**. *Then AN_0 is called* pushout complement *of l and m. Moreover, we obtain a unique AN' as pushout object of the pushout (2) in* **AHLNets**.

$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & I & \xrightarrow{\ r\ } & R \\ {\scriptstyle m}\downarrow & (1) & {\scriptstyle c}\downarrow & (2) & {\scriptstyle n}\downarrow \\ AN & \xleftarrow[d]{} & AN_0 & \dashrightarrow & AN' \end{array}$$

Proof. See [PER95]. □

Example 5 (Evolution of Apache Wave Platform (revisited))**.** The gluings of AHL-nets depicted in Fig. 6 describe a direct transformation of AHL-nets *Platform* ⇒ *Platform′* using production *insertLog* at match m. The diagram in Fig. 6a corresponds to the pushout (1) and the diagram in Fig. 6b corresponds to pushout (2) in Fact 3.

Now, we extend our framework to the gluing and transformation of AHL-process nets. For this purpose we define productions for AHL-process nets where the left hand and right hand side and the interface of the production are AHL-process nets.

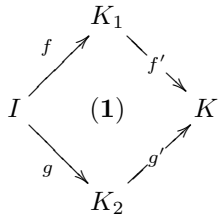
Definition 11 (Production for AHL-Process Nets)**.** A *production for AHL-process nets* $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ is a span of injective **AHLPNets**-morphisms $l : I \rightarrow L$ and $r : I \rightarrow R$.

$$L \xleftarrow{\ l\ } I \xrightarrow{\ r\ } R$$

[1] i. e. all elements in L that are mapped non-injectively by m
[2] i. e. all places in L that would leave a dangling arc, if deleted
[3] i. e. all elements in L that have a preimage in I

The following lemma states the fact that the gluing and the direct transformation of AHL-process nets via pushout constructions can be computed in the category of AHL-nets because every pushout in **AHLPNets** is also a pushout in **AHLNets**.

Lemma 2 (Pushout of AHL-Process Nets)**.** *Given AHL-process nets I, K_1 and K_2 and two AHL-net morphisms $f : I \to K_1$ and $g : I \to K_2$. If (1) is a pushout in* **AHLPNets** *then (1) is also pushout in* **AHLNets**.



Proof Idea. Constructing the pushout of the given span in the category **AHLNets** we obtain a pushout object K' together with a unique induced morphism $k : K' \to K$. Then by Lemma 1 the morphism k implies that K' is an AHL-process net leading to a unique morphism $k' : K \to K'$ by universal property of pushout (1). The morphisms k and k' can be shown to be inverse isomorphisms which by the uniqueness of pushouts implies that (1) is pushout in **AHLNets**. For a detailed proof see Appendix A.2. □

The gluing of AHL-nets may produce forward or backward conflicts as well as cycles in the causal relation. So for the gluing of two AHL-process nets via pushout construction the AHL-process nets have to be *composable* in order to obtain again an AHL-process net as a result of the gluing. Composability of AHL-process nets with respect to an interface means that the result of the gluing does not violate the process net properties in Def. 3.

A span of **AHLPNets**-morphisms $i_1 : I \to K_1$ and $i_2 : I \to K_2$ induces a causal relation between the elements of the interface I. This relation consists of the causal relation between elements in K_1 and K_2 and additionally between those elements in both of the AHL-process nets which is obtained by gluing over the interface.

Definition 12 (Induced Causal Relation)**.** Given three AHL-process nets I, K_1 and K_2, and two AHL-net morphisms $i_1 : I \to K_1$ and $i_2 : I \to K_2$. The *induced causal relation* $<_{(i_1,i_2)}$ is defined as the transitive closure of the relation $\prec_{(i_1,i_2)}$ defined by

$$\prec_{(i_1,i_2)} = \{(x,y) \in (P_I \uplus T_I) \times (P_I \uplus T_I) \mid i_1(x) <_{K_1} i_1(y) \text{ or } i_2(x) <_{K_2} i_2(y)\}.$$

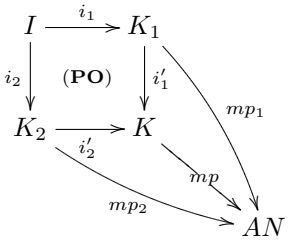
Definition 13 (Composability of AHL-Process Nets)**.** Given three AHL-process nets I, K_1 and K_2, and two AHL-net morphisms $i_1 : I \to K_1$ and $i_2 : I \to K_2$, where i_1 is injective. Then (K_1, K_2) *are composable w.r.t.* (I, i_1, i_2) if

1. *(No Cycles)* the induced causal relation $<_{(i_1,i_2)}$ is a strict partial order,
2. *(Non-Injective Gluing)*
   - for all $p_1 \neq p_2 \in IN(I)$ with $i_2(p_1) = i_2(p_2)$ there is $i_1(p_1) \in IN(K_1)$ or $i_1(p_2) \in IN(K_1)$,
   - for all $p_1 \neq p_2 \in OUT(I)$ with $i_2(p_1) = i_2(p_2)$: there is $i_1(p_1) \in OUT(K_1)$ or $i_1(p_2) \in OUT(K_1)$, and
3. *(No Conflicts)*
   - for all $p \in IN(I) : i_1(p) \notin IN(K_1) \Rightarrow i_2(p) \in IN(K_2)$,
   - for all $p \in OUT(I) : i_1(p) \notin OUT(K_1) \Rightarrow i_2(p) \in OUT(K_2)$.

The composability of AHL-process nets is a sufficient and necessary condition for the existence of the gluing of AHL-process nets as pushout in the category **AHLPNets**.

Fact 4 (Gluing of AHL-Process Nets)**.** *Given AHL-process nets I, K_1, K_2 and AHL-net morphisms $i_1 : I \to K_1$ and $i_2 : I \to K_2$ where i_1 is injective. Then there exists a pushout (PO) in the category* **AHLPNets** *(see Def. 3) iff (K_1, K_2) are composable w.r.t. (I, i_1, i_2). The AHL-process net K is then called gluing of K_1 and K_2 along i_1 and i_2, written $K = K_1 +_{(I,i_1,i_2)} K_2$.*

Extension to Processes. *In order to extend this gluing construction for AHL-processes in the category* **Proc**(**AN**) *(see Def. 5) one additionally requires AHL-morphisms $mp_1 : K_1 \to AN$ and $mp_2 : K_2 \to AN$ with $mp_1 \circ i_1 = mp_2 \circ i_2$. The pushout (PO) in* **AHLPNets** *then provides a unique morphism $mp : K \to AN$ such that (PO) is also a pushout in* **Proc**(**AN**)*.*



Proof-Idea. In order to show that the diagram (PO) constructed as pushout in **AHLNets** is also a pushout in the full subcategory **AHLPNets** $\subseteq$ **AHLNets** it suffices to show that the pushout object K is an AHL-process net. The fact that the gluing does not produce conflicts or cycles is ensured by the corresponding items 1 and 3 of the required composability of K_1 and K_2. Furthermore, item 2 ensures that there are no conflicts or violations of the unarity condition created by non-injective gluing.

The other way around the pushout (PO) in **AHLPNets** means that the pushout object K is an AHL-process net and by Lemma 2 (PO) is also a pushout in **AHLNets**. Then, it can be shown that the conditions of the composability can be derived from the fact that K satisfies the requirements of an AHL-process net .

For a detailed proof see Appendix A.3. □

We define a gluing relation for the transformation of AHL-process nets which is induced by a production ϱ for AHL-process nets and a match m. The gluing relation is a relation between the interface elements of ϱ which consists of the causal relation between elements in the codomain of m that are preserved by application of ϱ and the causal relation of the right hand side of the production, and additionally it consists of the causal relations that are obtained by gluing over the interface.

Definition 14 (Gluing Relation for Transformations)**.** Given a production for AHL-process nets $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ and a match $m : L \to K$ we define the relations

$$\prec_{(K,m)} = \{(x, y) \in (P_K \times (T_K \setminus m_T(T_L))) \uplus ((T_K \setminus m_T(T_L)) \times P_K) \mid x \in {\bullet y}\}$$

and $<_{(K,m)}$ as the transitive closure of $\prec_{(K,m)}$. Furthermore we define

$$\prec_{(\varrho,m)} = \{(x, y) \in (P_I \times T_I) \uplus (T_I \times P_I) \mid m \circ l(x) <_{(K,m)} m \circ l(y) \vee r(x) <_R r(y)\}$$

The transitive closure $<_{(\varrho,m)}$ of $\prec_{(\varrho,m)}$ is called *gluing relation* of production ϱ under match m.

For the transformation of AHL-process nets we define a *transformation condition* which is a necessary and sufficient condition that the direct transformation of an AHL-process nets exists. The satisfaction of the transformation condition by a production ϱ and a match m requires that the gluing condition for AHL-nets (see Def. 10) is satsfied. Moreover, it requires that the gluing condition is irreflexive and that the application of the production does neither produce any conflicts nor violates the unarity condition of AHL-process nets.

Definition 15 (Transformation Condition for AHL-Process Nets)**.** Given a production for AHL-process nets $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ and an AHL-process net K. Then ϱ satisfies the transformation condition under a (match) morphism $m : L \to K$ if

1. *(Gluing Condition)* the gluing condition is satisfied (see Def. 10),
2. *(No Cycles)* the gluing relation $<_{(\varrho,m)}$ of ϱ under m is a strict partial order,

3. *(Non-Injective Gluing)*
   - for all $p_1 \neq p_2 \in IN(I)$ with $m \circ l(p_1) = m \circ l(p_2)$ we have $r(p_1) \in IN(R)$ or $r(p_2) \in IN(R)$,
   - for all $p_1 \neq p_2 \in OUT(I)$ with $m \circ l(p_1) = m \circ l(p_2)$ we have $r(p_1) \in OUT(R)$ or $r(p_2) \in OUT(R)$,
4. *(No Conflicts)* for the sets of in and out places of the match

$$InP = \{x \in IN(I) \mid l(x) \in IN(L) \text{ and } m \circ l(x) \notin IN(K)\}, \text{ and}$$

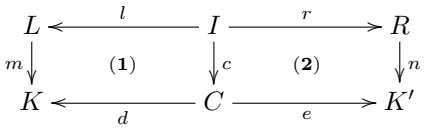
$$OutP = \{x \in OUT(I) \mid l(x) \in OUT(L) \text{ and } m \circ l(x) \notin OUT(K)\}$$

there is

$$r(InP) \subseteq IN(R) \text{ and } r(OutP) \subseteq OUT(R).$$

Theorem 1 (Direct Transformation of AHL-Process Nets)**.** *Given a production for AHL-process nets $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ and an AHL-process net K together with a morphism $m : L \rightarrow K$. Then the direct transformation of AHL-process nets with pushouts (1) and (2) in* **AHLPNets** *exists iff ϱ satisfies the transformation condition for AHL-process nets under m.*

Extension to processes. *In order to extend this construction for AHL-processes in the category* **Proc**(**AN**) *one additionally requires AHL-morphisms $mp : K \rightarrow AN$ and $rp : R \rightarrow AN$ with $mp \circ m \circ l = rp \circ r$. Then by composition of AHL-morphisms we obtain an AHL-process $cp = mp \circ d : C \rightarrow AN$ and the pushout (1) in* **AHLPNets** *is also a pushout of $mp \circ m$ and cp in* **Proc**(**AN**)*. Moreover, the pushout (2) in* **AHLPNets** *provides a unique morphism $mp' : K' \rightarrow AN$ such that mp' is pushout of cp and rp in* **Proc**(**AN**) *according to Fact 4.*



Proof-Idea. Satisfaction of the transformation condition for AHL-process nets means that the gluing condition for AHL-nets is satsfied which by Fact 3 implies that pushouts (1) and (2) can be constructed in **AHLNets**. It can be shown that the process net properties in Def. 5 are reflected by AHL-morphisms, implying that C is an AHL-process net and (1) is also a pushout in **AHLPNets**. Finally, it can be shown that the satisfaction of the transformation condition implies that C and R are composable w. r. t. (I, c, r), i. e. the pushout (2) in **AHLNets** is also a pushout in **AHLPNets**.

Vice versa, given pushouts (1) and (2) in **AHLPNets**, we have also pushouts in **AHLNets**, implying that the gluing condition for AHL-nets is satisfied. The satisfaction of the rest of the transformation condition can be obtained by composability of C and R w. r. t. (I, c, r) by pushout (2) in **AHLPNets**, and the construction of pushout complement C.

For a detailed proof see Appendix A.4. □

Example 6 (Evolution of Scenario Net)**.** The top row of Fig. 7 shows a production $\varrho : L \hookleftarrow I \hookrightarrow R$ for AHL-process nets that replaces two single invitations by one invitation of two users at a time. There is a match $m : L \rightarrow W$ into the AHL-process net W at the left bottom of Fig. 7 where the transitions are matched by inclusion and the places in the environments of the transitions are matched accordingly to the environments of the images in W. The match m satisfies the transformation condition for AHL-process nets. So the rule ϱ can be applied with match m, leading to the context net W_0 where the two invitations have been removed, and to the result net W' which is an AHL-process net containing the new transition *invite two*.

Note that there are three other possible matches for the rule ϱ into the AHL-process net W, but these matches violate the gluing condition for AHL-nets because in the case of all three matches the places w_4 and u_7 are identification points which are no gluing points (see Def. 10).

Moreover, consider a modified rule ϱ', containing places w_4 and u_7 in its interface and right-hand side. Then ϱ' and each of the possible matches satisfy the gluing condition for AHL-nets. But the

two matches m_1 and m_2, matching both transitions of L' to $invite_1$ respectively $invite_2$, violate the transformation condition for AHL-process nets. The reason for the violation is the second condition, i. e. for $u_6, u_8 \in IN(I)$ there is $m_1 \circ l(u_6) = m_1 \circ l(u_8)$, but there is neither $r(u_6)$ nor $r(u_8)$ in $IN(R)$. The same holds for m_2. Therefore, there exists no AHL-process net that is the result of a direct transformation via ϱ' and match m_1 respectively m_2.

5 Extension of Scenarios based on Platform Evolutions

In the previous section we have presented the rule-based transformation of AHL-nets and processes and we have shown how it can be used to evolve Apache Wave platforms and scenarios. As mentioned in Section 2 it is possible that the communication platform is modified at runtime and there may already exist some waves that correspond to the old version of the platform. So we have the case that there is an AHL-process $wave : Wave \to Platform$ and a direct transformation of AHL-nets $Platform \Rightarrow Platform'$. In this section we show under which condition the scenario $wave$ can be extended to a scenario $wave' : Wave \to Platform'$ of the new platform. We regard $wave'$ as an extension of $wave$ if the two processes "agree" in the context net of the direct transformation $Platform \Rightarrow Platform'$ in the following sense.

Definition 16 (Extension of AHL-Process based on AHL-Net Transformation)**.** Given an AHL-net AN and an AHL-process $mp : K \to AN$. Let $AN \overset{\varrho,m}{\Rightarrow} AN'$ be a direct transformation with pushouts (1) and (2) in **AHLNets** as depicted in Figure 8. Then we call $mp' : K \to AN'$ an *extension of mp* if there exists $mp_0 : K \to AN_0$ with $f \circ mp_0 = mp$ and $g \circ mp_0 = mp'$.

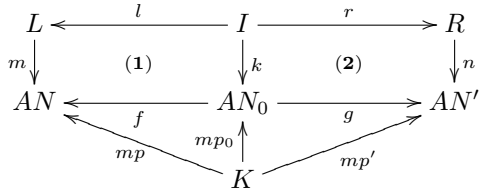


Figure 8: Extension of AHL-Process

The following extension condition is a sufficient and necessary condition for the extension mp' of an AHL-process mp based on an AHL-net transformation. In order to satisfy the extension condition, the transformation must not delete any place or transition that have an occurrence in the AHL-process mp.

Definition 17 (Extension Condition)**.** Given an AHL-net AN, an AHL-process $mp : K \to AN$ and a direct transformation $AN \overset{\varrho,m}{\Rightarrow} AN'$. We define the the set PP of process points as

$$PP = \{x \in P_L \mid \exists p \in P_K : mp_P(p) = m_P(x)\} \cup \{x \in T_L \mid \exists t \in T_K : mp_T(t) = m_T(x)\}$$

We say that mp and ϱ, m satisfy the extension condition if all process points are gluing points (see Def. 10), i. e. $PP \subseteq GP$.

Theorem 2 (Extension of AHL-Process based on AHL-Net Transformation)**.** *Given an AHL-net AN, an AHL-process $mp : K \to AN$ and a direct transformation $AN \overset{\varrho,m}{\Rightarrow} AN'$ with pushouts (1) and (2) in* **AHLNets** *as depicted in Figure 8. There exists an extension $mp' : K \to AN'$ of mp if and only if mp and ϱ, m satisfy the extension condition.*

Proof-Idea. If the extension condition is satisfied, it can be explicitely shown that there exists a well-defined morphism $mp_0 : K \to AN_0$ defined by $mp_0 = f^{-1} \circ mp$. Then the extension $mp' : K \to AN'$ is obtained by composition $mp' = g \circ mp_0$, satisfying the required properties.

Vice versa, the existence of an extension $mp' : K \to AN'$ implies the existence of a suitable morphism $mp_0 : K \to AN_0$ which can be used to show that all process points are gluing points.

For a detailed proof see Appendix A.5. □

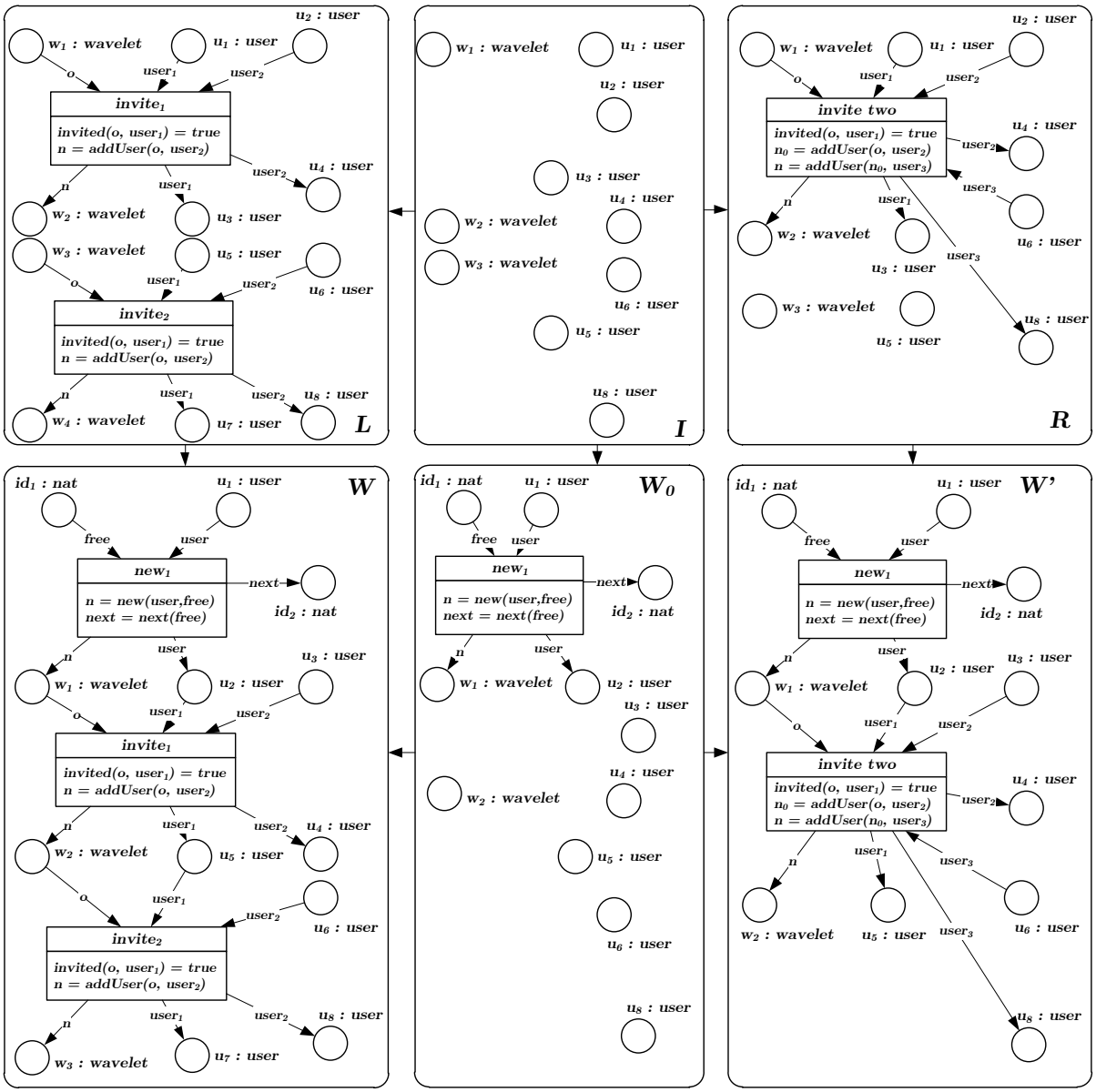


Figure 7: Evolution $W \overset{o,m}{\Rightarrow} W'$ of Scenario Nets

Example 7 (Extension of Scenario based on Platform Evolution)**.** Consider again the AHL-process net W in the left bottom of Fig. 7 together with an AHL-morphism $mp : W \to \mathit{Platform}$ matching all elements in W to the corresponding elements with similar names. Furthermore, consider the AHL-net transformation $\mathit{Platform} \Rightarrow \mathit{Platform}'$ via production *insertLog* and match m in Example 4. The set of process points with this transformation and AHL-process is the set $PP = \{u_1, u_2, w_1, w_2\}$ which corresponds exactly to the set of gluing points and therefore $PP \subseteq GP$. So the AHL-process mp can be extended to a process $mp' : W \to \mathit{Platform}'$ that maps all elements in the same way as mp and that corresponds to a scenario in the modified platform In contrast, for the AHL-process $\mathit{wave} : \mathit{Wave} \to \mathit{Platform}$ in Example 2 the set of process points $PP = \{u_1, u_2, w_1, w_2, \mathit{modify\ text}\}$ and we have $PP \not\subseteq GP$ because *modify text* is not a gluing point. Thus, there exists no extension of *wave* based on the platform evolution $\mathit{Platform} \Rightarrow \mathit{Platform}'$.

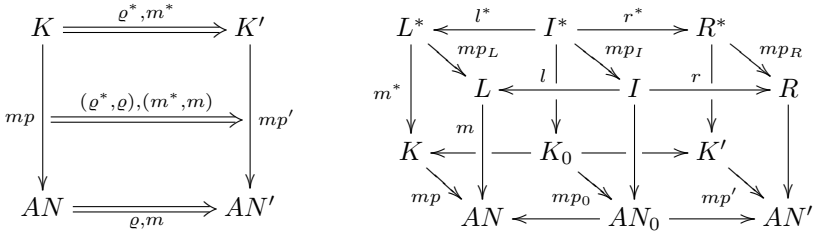
6 Evolution of Scenarios based on Platform Evolutions

As we have seen in Example 7 there are cases of scenarios and platform evolutions that do not satisfy the extension condition. The reason in the discussed example is that the scenario *wave* contains three occurrences of the action *modify text*, but there is no corresponding action in the new platform $\mathit{Platform}'$. Nonetheless, the feature to modify some text in a wavelet has not been fully removed from the communication platform, but it has been replaced by the new action *modify and log* which does more or less the same as the old action with the only difference that it does additionally create a log entry. So as discussed at the end of Section 2 an intuitive solution is to apply the modification of the platform also to the scenario *wave*, leading to a scenario wave' as depicted in Fig. 4 where all occurrences of the action *modify text* have been replaced by the new version *modify and log* of the action.

In this section we give a general construction for the modification of scenarios based on a special kind of platform evolution, replacing one single action at a time. For this purpose, since scenarios are modeled as AHL-processes, we need productions and the direct transformation of AHL-processes as defined in the following.

Definition 18 (Production for AHL-Processes)**.** A *production for AHL-processes* is a span $(\varrho^*, \varrho) : mp_L \overset{(l^*,l)}{\longleftarrow} mp_I \overset{(r^*,r)}{\longrightarrow} mp_R$ of injective **AHLProcs**-morphisms as shown in Figure 9b.

Definition 19 (Direct Transformation of AHL-Processes)**.** Given a production $\varrho : mp_L \overset{(l^*,l)}{\longleftarrow} mp_I \overset{(r^*,r)}{\longrightarrow} mp_R$ for AHL-processes and a (match) morphism $(m^*, m) : mp_L \to mp$. Then a direct transformation $mp \overset{(\varrho^*,\varrho),(m^*,m)}{\Longrightarrow} mp'$ is given by the commuting cube in Figure 9b where the front and back faces are pushouts in **AHLNets** and **AHLPNets**, respectively.



(a) Transformation of AHL-process

(b) Commuting Cube

Figure 9: Transformation of AHL-process

In the following we show how to construct productions for processes from a special type of production for AHL-nets, called action evolution. An action evolution is a direct transformation of AHL-nets that uses a special kind of production. The main aspect of such a production is that it contains exactly one transition in its left-hand side.

Definition 20 (Action Evolution)**.** A production $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ for AHL-process nets is called a *production for an action evolution* if

1. *(Single Action)* L contains only one transition and its environment, i. e. $T_L = \{t_\varrho\}$ and for all $p \in P_L$: $p \in {}^\bullet t_\varrho \cup t_\varrho {}^\bullet$,

2. *(Unique Arc Inscriptions)* all arcs in one direction have different inscriptions, i. e. $(term_1, p_1) \oplus (term_2, p_2) \leq pre_L(t_\varrho)$ implies $term_1 \neq term_2$ and $p_1 \neq p_2$, and the same holds for post arcs[4],

3. *(Preserved Environment)* ϱ is non-deleting on places, i. e. $P_L = l_P(P_I)$, and

4. *(Preserved Input and Output)* ϱ preserves input and output places, i. e. for all $p \in P_I$:

   - $l(p) \in IN(L) \Rightarrow r(p) \in IN(R)$ and
   - $l(p) \in OUT(L) \Rightarrow r(p) \in OUT(R)$.

Given an AHL-net AN and a match $m : L \to AN$, a direct transformation $AN \overset{\varrho,m}{\Longrightarrow} AN'$ is called *action evolution*.

Example 8 (Action Evolution)**.** The production *insertLog* in Fig. 5 is a production for action evolution. The left-hand side of *insertLog* consists of only one transition *modify text* and its environment. The inscriptions of pre respectively post arcs of the left-hand side are unique. Note that the fact that there are similar arc inscriptions *user* on pre and post arcs does not violate the single action condition. Moreover, the production is non-deleting on places and all input respectively output places of the left-hand side are also input respectively output places in the right-hand side of the production.

Hence, the direct transformation *Platform* $\Rightarrow$ *Platform*$'$ via production *insertLog* and match m shown in Fig. 6 is an action evolution.

Now, the following theorem states that for every process $mp : K \to AN$ and an action evolution of the net AN there exists a corresponding transformation of AHL-processes $mp \Rightarrow mp'$. As result we obtain a process corresponding to the result of the action evolution, where all occurrences of the modified part in AN have been modified in K as well.

Theorem 3 (Process Evolution based on Action Evolution)**.** *Given an action evolution $AN \overset{\varrho,m}{\Longrightarrow} AN'$ via production $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$, and a process $mp : K \to AN$. Then there exists a production (ϱ^+, ϱ) for AHL-processes and a direct transformation $mp \overset{(\varrho^+,\varrho)}{\Longrightarrow} mp'$ as depicted in Figure 10a that realizes the changes described by ϱ on all occurrences in the process mp.*
CONSTRUCTION: *Let $(m_i : L \to K)_{i \in \mathcal{I}}$ be the class of all matches $m_i : L \to K$ with $mp \circ m_i = m$.*

1. *The production for AHL-process nets $\varrho^+ : L^+ \overset{l^+}{\leftarrow} I^+ \overset{r^+}{\rightarrow} R^+$ is defined as componentwise coproduct in* **AHLPNets***:*

   - $X^+ = \coprod_{i \in \mathcal{I}} X$ *with injections* $\iota_i^X : X \to X^+$ *for* $X \in \{L, I, R\}$,
   - $x^+ = \coprod_{i \in \mathcal{I}} x$ *for* $x \in \{l, r\}$

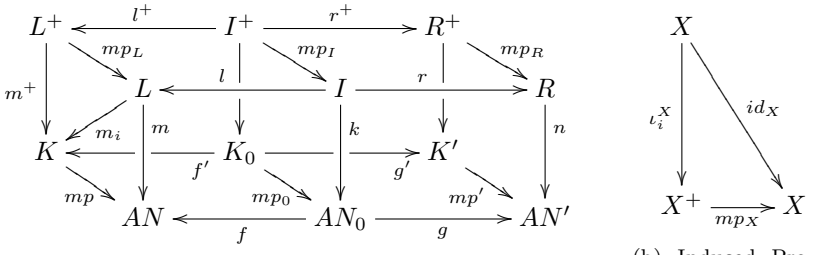
2. *The processes $mp_X : X^+ \to X$ for $X \in \{L, I, R\}$ are the unique induced morphisms with $mp_X \circ \iota_i^X = id_X$ for all $i \in \mathcal{I}$ (see Figure 10b).*

3. *The match $m^+ : L^+ \to K$ is the unique induced morphism with $m^+ \circ \iota_i^L = m_i$ for all $i \in \mathcal{I}$.*

4. *K_0 and K' are constructed as direct AHL-process net transformation in the back of Figure 10a.*

5. *$mp_0 : K_0 \to AN_0$ is defined as $mp_0 = f^{-1} \circ mp \circ f'$, and $mp' : K' \to AN'$ is induced by the right pushout in the back of Figure 10a.*

[4]This condition is necessary in order avoid that a match of the rule can correspond ambiguously to one occurrence in a process of a matched AHL-net. For the modelling with AHL-nets equal term inscriptions in the environment of one transition are not really necessary, since equality of two terms can also be expressed as a condition of the transition.



(a) Process Evolution

(b) Induced Process

Figure 10: Process evolution based on action evolution

Proof-Idea. The construction of ϱ^+ by coproducts in **AHLPNets** as given above is well-defined, and the universal property of coproducts can be used to show that (ϱ^+, m^+) is a production for AHL-processes (see Def. 18). Furthermore, the construction of L^+ induces a unique $m^+ : L^+ \to K$ with $m^+ \circ \iota_i^L = m_i$ for all $i \in \mathcal{I}$, and by compatibility of m_i with m for all $i \in \mathcal{I}$ it can be concluded that (m^+, m) is an **AHLProcs**-morphism.

The existence of the direct transformation of AHL-process nets with pushouts in the back of Fig. 10a can be shown using the properties required for action evolutions in Def. 20.

Finally, it can be shown that there exists a well-defined AHL-morphism $mp_0 : K_0 \to AN_0$ defined as $mp_0 = f^{-1} \circ mp \circ f'$, leading to a unique morphism $mp' : K' \to AN'$ induced by the pushout in the right back of Fig. 10a such that all diagrams in the cube commute.

For a detailed proof see Appendix A.6. □

Example 9 (Evolution of Scenario based on Platform Evolution)**.** Now, consider again the platform evolution *Platform* ⇒ *Platform′* via production *insertLog* (see Fig. 5) and match m in Example 4 and the scenario $wave : Wave \to Platform$ (see Fig. 2) in Example 2. The platform evolution via production *insertLog* is an action evolution and there are three possible matches $m_i : L \to Wave$ consistent with m, mapping *modify text* to the three occurrences of the transition in *Wave*. We can construct a production $insertLog^+$ consisting of three copies of the left-hand side, interface and right-hand side of $insertLog^+$ as depicted in Fig. 11. Moreover, we obtain processes $mp_L : L^+ \to L$, $mp_I : I^+ \to I$ and $mp_R : R^+ \to R$ mapping every copy to its original, and there is a match $m^+ : L^+ \to Wave$ that maps all copies according to the matches $m_i : L \to Wave$. By application of $insertLog^+$ with match m^+ we obtain the scenario $wave' : Wave' \to Platform'$ as depicted in Fig. 11 where all occurrences of the *modify text* action have been replaced by a *modfiy and log* action.

In the future we will consider different types of multiple action evolutions, i. e. evolutions that modify more than one transition at once. Two promising approaches are *forward* and *backward multiple action evolutions*, which are evolutions where we have additional information about the relation between the left-hand side L and right-hand side R of the used production. These relations are expressed as morphisms between the "skeletons" $Skel(L)$ and $Skel(R)$, that is only the low-level Petri net structure without any high-level data part, of L respectively R.

A forward multiple action evolution consists of a morphism $\phi : Skel(L) \to Skel(R)$. The morphism describes that an element x is replaced by $\phi(x)$. This allows for example to express that one new action *modify text* is the new version of multiple old actions *insert text* and *remove text*. The corresponding process evolution should then replace all occurrences of *insert text* as well as of *remove text* by new occurrences of *modify text*.

A backward multiple action evolution consists of a morphism $\psi : Skel(R) \to Skel(L)$, describing that an element x replaces $\psi(x)$. This allows for example to express that a new action *modify without invitation* is the new version of an old action *modify text* and an existing action *invite user* is removed without any replacement. The corresponding process evolution should then replace all occurrences of *modify text* while deleting all occurrences of *invite user*.

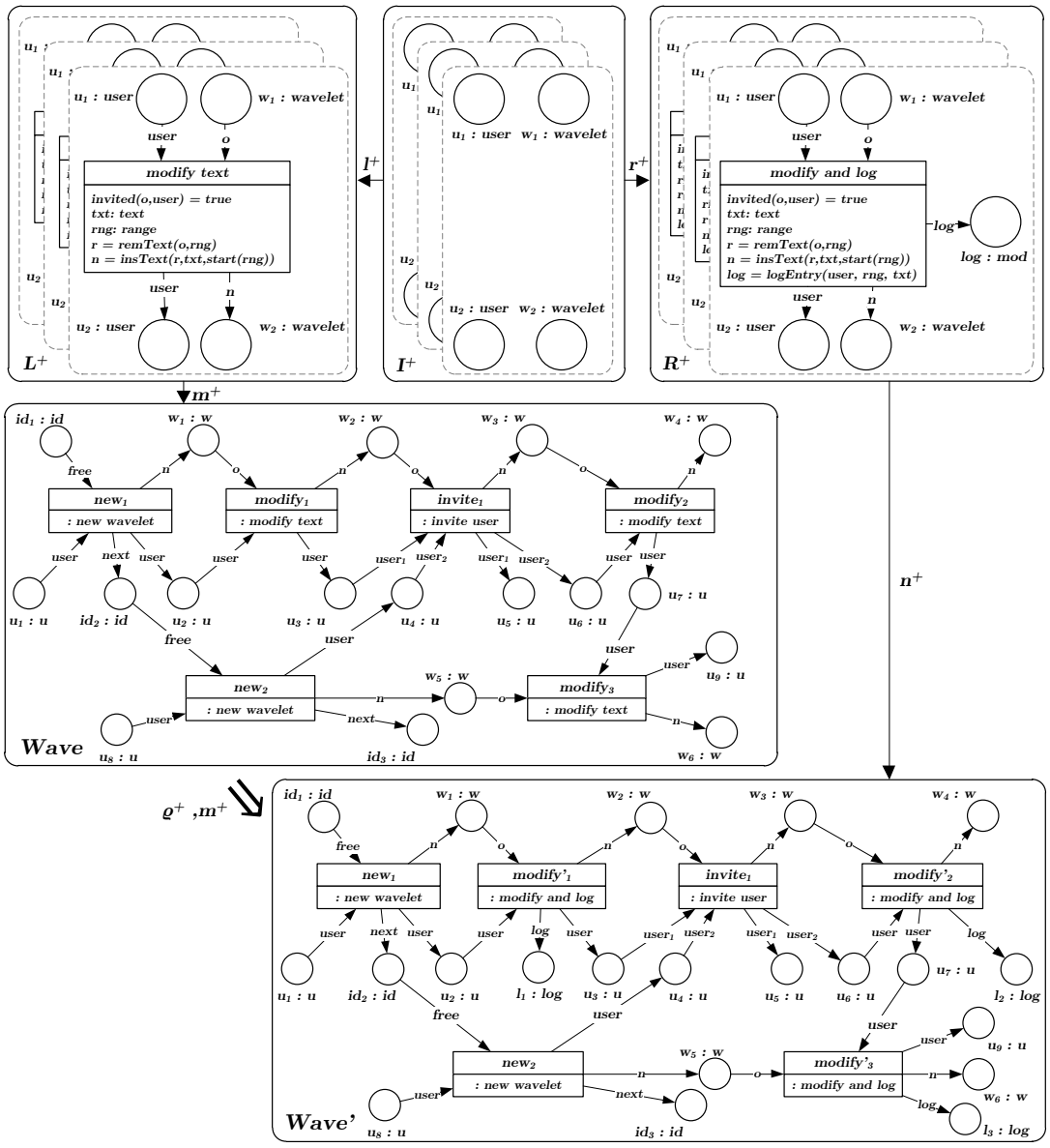


Figure 11: Process evolution based on action evolution

7 Conclusion

Algebraic high-level (AHL) nets are a well-known modelling technique based on Petri nets [Pet62, Rei85, Roz87] with algebraic data types [EM85]. In this paper we have shown that AHL nets, AHL processes, and AHL transformations can be considered as integrated framework for modelling the evolution of communication platforms. In previous papers it was shown already how to use this framework to model communication platforms like Skype [HM10] and Google Wave [EG11]. In this paper we have used the evolution of Apache Wave platforms and scenarios as running example, where platforms are modelled by AHL-nets and scenarios by AHL-processes. The evolution on both levels is defined by rule-based modifications in the sense of graph transformation systems [EEPT06]. While transformations of AHL-nets are introduced already in [PER95] the corresponding problem for AHL-processes is much more difficult as shown in Section 4.

The first main result shows under which conditions AHL-net processes can be extended if the corresponding AHL-net is transformed. This result can be applied to show the extension of scenarios for a given platform evolution. The second main result shows how AHL-net processes can be transformed based on a special kind of transformation for AHL-nets, corresponding to action evolution of platforms. In future work we will study the case of multiple action evolution, which is only briefly discussed in this paper. Moreover we will analyse what kind of properties can be preserved by evolution of platforms and scenarios.

A Detailed Proofs

A.1 Proof of Lemma 1 (AHL-Morphisms Reflect AHL-Process Nets)

Given an AHL-morphism $f : K_1 \to K_2$. If K_2 is an AHL-process net then also K_1.

Proof. Given AHL-morphism $f : K_1 \to K_2$ with AHL-process net K_2. In order to show that K_1 is an AHL-process net we have to show that it is unary, there are no forward or backward conflicts and the causal relation $<_{K_1}$ is a strict partial order.

Unarity. Let us assume that K_1 is not unary, i. e. there are $p \in P_{K_1}$, $t \in T_{K_1}$ with

$$(term_1, p) \oplus (term_2, p) \leq pre_{K_1}(t) \text{ or } (term_1, p) \oplus (term_2, p) \leq post_{K_1}(t)$$

Let $(term_1, p) \oplus (term_2, p) \leq pre_{K_1}(t)$.
Since AHL-morphisms preserve pre conditions there is

$$(id_{T_{OP}(X)} \otimes f_P)^{\oplus} \circ pre_{K_1}(t) = pre_{K_2}(f_T(t))$$

and hence

$$\begin{aligned}(term_1, f_P(p)) \oplus (term_2, f_P(p)) &= (id_{T_{OP}(X)} \otimes f_P)^{\oplus}((term_1, p) \oplus (term_2, p)) \\ &\leq pre_{K_2}(f_T(t))\end{aligned}$$

This implies that K_2 is not unary, contradicting the fact that K_2 is an AHL-process net.
The case that $(term_1, p) \oplus (term_2, p) \leq post_{K_1}(t)$ works analogously. Hence K_1 is unary.

No forward conflict. Let us assume that K_1 has a forward conflict, i. e. there is $p \in P_{K_1}$, $t_1 \neq t_2 \in T_{K_1}$ with $p \in \bullet t_1 \cap \bullet t_2$. This means that there are $term_1, term_2 \in T_{OP}(X)_{type(p)}$ such that

$$(term_1, p) \leq pre_{K_1}(t_1) \text{ and } (term_2, p) \leq pre_{K_1}(t_2)$$

and since AHL-morphisms preserve pre and post conditions we obtain

$$\begin{aligned}(term_1, f_P(p)) &= (id_{T_{OP}(X)} \otimes f_P)^{\oplus}(term_1, p) \\ &\leq pre_{K_2}(f_T(t_1))\end{aligned}$$

and

$$\begin{aligned}(term_2, f_P(p)) &= (id_{T_{OP}(X)} \otimes f_P)^{\oplus}(term_2, p) \\ &\leq pre_{K_2}(f_T(t_2))\end{aligned}$$

In the case that $f_T(t_1) \neq f_T(t_2)$ the fact that $f_P(p) \in \bullet f_T(t_1) \cap \bullet f_T(t_2)$ means that K_2 has a forward conflict, contradicting the fact that K_2 is an AHL-process net. So let us consider the fact that $f_T(t_1) = t = f_T(t_2)$. Then we have

$$(term_1, f_P(p)) \oplus (term_2, f_P(p)) \leq t$$

which contradicts the fact that K_2 is unary. Hence K_1 has no forward conflict.

No backward conflict. The proof for this case works analogously to the one for forward conflicts because AHL-morphisms preserve post as well as pre conditions and K_2 has no backward conflicts.

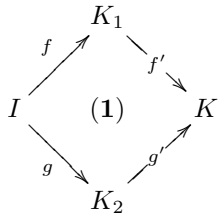
Strict partial order. We have to show that $<_{K_1}$ is irreflexive. So, let us assume that $<_{K_1}$ is not irreflexive, i. e. there exists a cycle $x <_{K_1} x$. This implies $f(x) <_{K_2} f(x)$ because AHL-morphisms preserve pre and post conditions. This contradicts the fact that $<_{K_2}$ is irreflexive because it is an AHL-process net.

Hence, also $<_{K_1}$ is irreflexive.

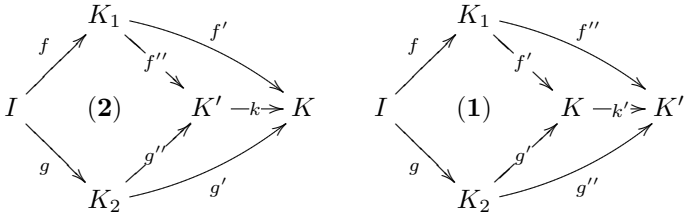
□

A.2 Proof of Lemma 2 (Pushout of AHL-Process Nets)

Given AHL-process nets I, K_1 and K_2 and two AHL-net morphisms $f : I \to K_1$ and $g : I \to K_2$. If (1) is a pushout in **AHLPNets** then (1) is also pushout in **AHLNets**.



Proof. Since the category **AHLNets** has pushouts we obtain pushout (2) in **AHLNets**. Then by the fact that **AHLPNets** is a subcategory of **AHLNets** by the commutativity of (1) we obtain a unique morphism $k : K' \to K$ with $k \circ f'' = f'$ and $k \circ g'' = g'$.



By Lemma 1 we have that K' is an AHL-process net and by the fact that **AHLPNets** is full subcategory of **AHLNets** the morphisms f'' and g'' become **AHLPNets**-morphisms. So the commutativity of (2) by the universal property of pushout (1) in **AHLPNets** implies a unique morphism $k' : K \to K'$ with $k' \circ f' = f''$ and $k' \circ g' = g''$. Now we have

$$k \circ k' \circ f' = k \circ f'' = f' \quad \text{and} \quad k \circ k' \circ g' = k \circ g'' = g'$$

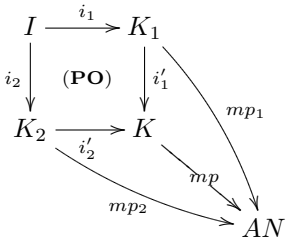
which by the universal property of pushout (1) implies that $k \circ k' = id_K$. Analogously we obtain by the universal property of pushout (2) that $k' \circ k = id_{K'}$ and, thus, k and k' become inverse isomorphisms. Hence, by the uniqueness of pushouts up to isomorphism it follows that (1) is also pushout in **AHLNets**.

□

A.3 Proof of Fact 4 (Gluing of AHL-Process Nets)

Given AHL-process nets I, K_1, K_2 and AHL-net morphisms $i_1 : I \to K_1$ and $i_2 : I \to K_2$ where i_1 is injective. Then there exists a pushout (PO) in the category **AHLPNets** (see Def. 3) iff (K_1, K_2) are composable w. r. t. (I, i_1, i_2). The AHL-process net K is then called gluing of K_1 and K_2 along i_1 and i_2, written $K = K_1 +_{(I,i_1,i_2)} K_2$.

Extension to Processes. In order to extend this gluing construction for AHL-processes in the category **Proc**(**AN**) (see Def. 5) one additionally requires AHL-morphisms $mp_1 : K_1 \to AN$ and $mp_2 : K_2 \to AN$ with $mp_1 \circ i_1 = mp_2 \circ i_2$. The pushout (PO) in **AHLPNets** then provides a unique morphism $mp : K \to AN$ such that (PO) is also a pushout in **Proc**(**AN**).



Proof. We show the two directions of the proof seperately.

If. Given the AHL-process nets K_1, K_2 and I and morphisms i_1, i_2 as above we construct the pushout (PO) in the category **AHLNets**.

In order to show that (PO) is also a pushout in the full subcategory **AHLPNets** it suffices to show that the AHL-net K is an AHL-process net, i. e. K is unary, it has no forward or backward conflicts, and the causal relation $<_K$ is a strict partial order.

Unarity. Let us assume that K is not unary, i. e. there are $p \in P_K$, $t \in T_K$ with

$$(term_1, p) \oplus (term_2, p) \leq pre_K(t) \quad \text{or} \quad (term_1, p) \oplus (term_2, p) \leq post_K(t)$$

Let us consider the case that $(term_1, p) \oplus (term_2, p) \leq pre_K(t)$. Due to the universal property of pushout (PO) there is $a \in \{1, 2\}$ and $t' \in T_{K_a}$ with $i'_{a,T}(t') = t$ and since AHL-morphisms preserve pre and post conditions there is

$$pre_K(t) = (id_{T_{OP}(X)} \otimes i'_{a,P})^{\oplus} \circ pre_{K_a}(t').$$

So the fact that $(term_1, p) \oplus (term_2, p) \leq pre_K(t)$ implies

$$(term_1, p_1) \oplus (term_2, p_2) \leq pre_{K_a}(t')$$

with $i'_a(p_1) = p = i'_a(p_2)$.

- **Case 1.** There is $a = 1$.
  The fact that $i'_1(p_1) = p = i'_1(p_2)$ by pushout (PO) implies that there are $x_1, x_2 \in P_I$ with $i_1(x_1) = p_1$, $i_1(x_1) = p_2$ and $i_2(x_1) = i_2(x_2)$. Moreover, $(term_1, p_1) \oplus (term_2, p_2) \leq pre_{K_1}(t')$ means $i_1(x_1), i_1(x_2) \notin OUT(K_1)$.
  - **Case 1.1.** There are $x_1, x_2 \in OUT(I)$.
    Then by composability of K_1 and K_2 w. r. t. (I, i_1, i_2) we have that $i_1(x_1) \in OUT(K_1)$ or $i_1(x_2) \in OUT(K_1)$, contradicting the fact that $i_1(x_1), i_1(x_2) \notin OUT(K_1)$.
  - **Case 1.2.** There is $x_1 \notin OUT(I)$.
    This means that there is $t_0 \in T_I$ with $(term_0, x_1) \leq pre_I(t_0)$. By the fact that AHL-morphisms preserve pre conditions, we obtain $(term_0, i_1(x_1)) \leq pre_{K_1}(i_1(t_0))$. This implies that $i_1(t_0) = t'$ because K_1 is an AHL-process net which does not have any forward conflicts. Moreover, we have $term_0 = term_1$. So, using again the fact that AHL-morphisms preserve pre conditions, we obtain $(term_2, x_2) \in pre_I(t_0)$ and furthermore $(term_1, i_2(x_1)) \oplus (term_2, i_2(x_2)) \leq pre_{K_2}(i_2(t_0))$. Hence, by the fact that $i_2(x_1) = i_2(x_2)$ we have that K_2 is not unary. This is a contradiction because K_2 is an AHL-process net.

  - **Case 1.3.** There is $x_2 \notin OUT(I)$.
    Analogously to Case 1.2 this case leads to a contradiction because AHL-morphisms preserve post as well as pre conditions and AHL-process net K_1 does also have no backward conflicts.

- **Case 2.** There is $a = 2$.
  Since i_1 is injective, also i'_2 is injective, because injective AHL-morphisms are closed under pushouts. Thus, we have $p_1 = p_2$, which means that K_2 is not unary. This is a contradiction to the fact that K_2 is an AHL-process net.

The case $(term_1, p) \oplus (term_2, p) \leq post_K(t)$ works analogously. Hence, K is unary.

No forward conflicts. Let us assume that K has a forward conflict, i. e. there are $p \in P_K$ and $t_1 \neq t_2 \in T_K$ with $p \in {\bullet}t_1 \cap {\bullet}t_2$.

- **Case 1:** There is $a \in \{1, 2\}$ such that $t_1, t_2 \in i'_{a,T}(T_{K_a})$.
  Then we have $t'_1 \neq t'_2 \in T_{K_a}$ with

  $$i'_{a,T}(t'_1) = t_1 \quad \text{and} \quad i'_{a,T}(t'_2) = t_2$$

  and there are $p_1, p_2 \in P_{K_a}$ with

  $$i'_{a,P}(p_1) = p = i'_{a,P}(p_2) \quad \text{and} \quad p_1 \in {\bullet}t'_1, p_2 \in {\bullet}t'_2$$

  because AHL-morphisms preserve pre conditions.

- **Case 1.1** $p_1 = p_2$.
  This means that K_a has a forward conflict which contradicts the fact that K_a is assumed to be an AHL-process net.

- **Case 1.2** $p_1 \neq p_2$.
  Since i'_2 is injective, this implies that $a = 1$. Then, $i'_1(p_1) = p = i'_1(p_2)$ implies $x_1, x_2 \in P_I$ with $i_1(x_1) = p_1$, $i_1(x_2) = p_2$ and $i_2(x_1) = i_2(x_2)$. Moreover, $i_1(x_1) = p_1 \in {\bullet}t'_1$ means that $p_1 \notin OUT(K_1)$, and $i_1(x_2) = p_2 \in {\bullet}t'_2$ means that $p_2 \notin OUT(K_1)$.

  Case 1.2.1. There is $x_1, x_2 \in OUT(I)$.
  Then by composability of K_1 and K_2 w. r. t. (I, i_1, i_2) it follows that $i_1(x_1) \in OUT(I)$ or $i_1(x_2) \in OUT(K_1)$, contradicting the fact that $i_1(x_1), i_1(x_2) \notin OUT(K_1)$.

  Case 1.2.2. There is $x_1 \notin OUT(I)$, $x_2 \in OUT(I)$.
  By the fact that $x_2 \in OUT(I)$ and $i_1(x_2) \notin OUT(K_1)$ the composability of K_1 and K_2 w. r. t. (I, i_1, i_2) implies $i_2(x_2) \in OUT(K_2)$.
  Furthermore, the fact that $x_1 \notin OUT(I)$ means that there is some $t_0 \in T_I$ with $(term_0, x_1) \leq pre_I(t_0)$. By the fact that AHL-morphisms preserve pre conditions, we obtain $(term_0, i_1(x_1)) \leq pre_{K_1}(i_1(t_0))$. This implies that $i_1(t_0) = t'_1$ because K_1 is an AHL-process net which does not have any forward conflicts. Moreover, we have $term_0 = term_1$. Using again the fact that AHL-morphisms preserve pre conditions, we obtain $(term_1, i_2(x_1)) \in pre_{K_2}(i_2(t_0))$ which means that $i_2(x_1) = i_2(x_2) \notin OUT(K_2)$, contradicting the fact that $i_2(x_2) \in OUT(K_2)$.

  Case 1.2.3. There is $x_1 \in OUT(I)$, $x_2 \notin OUT(I)$.
  This case is similar to Case 1.2.2.

  Case 1.2.4. There is $x_1, x_2 \notin OUT(I)$. Then we have $t_0, t'_0 \in T_I$ with $(term_0, x_1) \leq pre_I(t_0)$ and $(term'_0, x_2) \leq pre_I(t'_0)$. Analogously to Case 1.2.2 we obtain that $i_1(t_0) = t'_1$ and $i_1(t'_0) = t'_2$, and using the fact that AHL-morphisms preserve pre conditions, we have $i_2(x_1) \leq pre_{K_2}(i_2(t_0))$ and $i_2(x_1) = i_2(x_2) \leq pre_{K_2}(i_2(t'_0))$. Since K_2 is an AHL-process net, it does not have any forward conflicts, implying that $i_2(t_0) = i_2(t'_0)$. Thus, we have

  $$t_1 = i'_1(t'_1) = i'_1(i_1(t_0)) = i'_2(i_2(t_0)) = i'_2(i_2(t'_0)) = i'_1(i_1(t'_0)) = i'_1(t'_2) = t_2$$

  which contradicts the fact that $t_1 \neq t_2$.

- **Case 2:** There is $t_1 \in i_1(T_{K_1})$ and $t_2 \in i_2(T_{K_2})$.
  Then we have $t_1' \in T_{K_1}, t_2' \in T_{K_2}$ with

  $$i_1'(t_1') = t_1 \quad \text{and} \quad i_2'(t_2') = t_2$$

  and since AHL-morphisms preserve pre conditions there are $p_1 \in P_{K_1}$, $p_2 \in P_{K_2}$ with

  $$i_1'(p_1) = p,\ p_1 \in {}^\bullet t_1' \quad \text{and} \quad i_2'(p_2) = p,\ p_2 \in {}^\bullet t_2'.$$

  By the fact that K is a pushout object of (PO) this implies a place $p_0 \in P_I$ with

  $$i_1(p_0) = p_1 \quad \text{and} \quad i_2(p_0) = p_2.$$

- **Case 2.1:** There is $p_0 \in OUT(I)$.
  Due to the fact that $p_1 \in {}^\bullet t_1'$, we have $i_1(p_0) \notin OUT(K_1)$, which by the composability of (K_1, K_2) w. r. t. (I, i_1, i_2) implies that $i_2(p_0) \in OUT(K_2)$ contradicting the fact that $i_2(p_0) = p_2 \in {}^\bullet t_2'$.
- **Case 2.2:** There is $p_0 \notin OUT(I)$.
  This means that there is $t_0 \in T_I$ with $p_0 \in {}^\bullet t_0$. By the fact that i_1 is an AHL-morphism which preserves pre conditions we have $p_1 \in {}^\bullet i_1(t_0)$ which together with the fact that $p_1 \in {}^\bullet t_1'$ means that $i_1(t_0) = t_1'$ because K_1 has no forward conflicts. Analogously, due to the fact that also K_2 has no forward conflict we obtain that $i_2(t_0) = t_2'$. Thus, by commutativity of (PO) we have

  $$t_1 = i_1'(t_1') = i_1'(i_1(t_0)) = i_2'(i_2(t_0)) = i_2'(t_2') = t_2$$

  which contradicts the assumption that $t_1 \neq t_2$.

Hence, all cases lead to a contradiction which means that K has no forward conflict.

No backward conflicts. The proof that K does not have any backward conflicts works analogously to the proof concerning the forward conflicts, because AHL-morphisms preserve post as well as pre conditions, and the definition of the composability consists of corresponding conditions for input as well as for output places.

Strict partial order. Since $<_K$ is defined as a transitive closure, it suffices to show that e have to show that $<_K$ is irreflexive. Due to the fact that AHL-morphisms preserve pre and post conditions we obtain the causal relation of $<_K$ as the transitive closure of

$$\bigcup_{a \in \{1,2\}} \{(i_a'(x), i_a'(y)) \mid x, y \in P_{K_a} \uplus T_{K_a}, x <_{K_a} y\}$$

This means that elements which are causally related in K_1 or K_2 are also causally related in K. Additionally it is possible that elements in the net K are related due to the gluing of one or more elements.
Moreover, if for two interface elements $x_0, y_0 \in P_I \uplus T_I$ the images of these elements are causally related in K, i. e. we have the following **Statement (A)**:

$$\forall x_0, y_0 \in P_I \uplus T_I : i_1'(i_1(x_0)) <_K i_1'(i_1(y_0)) \Rightarrow x_0 <_{(i_1,i_2)} y_0$$

We prove this statement because we need it in the following:
Let $x_0, y_0 \in P_I \uplus T_I$ with $i_1'(i_1(x_0)) <_K i_1'(i_1(y_0))$. Then there is $a \in \{1,2\}$ such that either there is $i_a(x_0) <_{K_a} i_a(y_0)$ or there is $z_0 \in P_I \uplus T_I$ with $i_a(x_0) <_{K_a} i_a(z_0)$ and $i_1'(i_1(x_0)) <_K i_1'(i_1(z_0)) <_K i_1'(i_1(y_0))$. This recursively leads to the fact that $x_0 <_{(i_1,i_2)} y_0$ because the induced causal relation is transitive.

Let us now assume that $<_K$ is not irreflexive, i. e. there exists $x \in P_K \uplus T_K$ s.t. $x <_K x$.

- **Case 1.** There is no element $z \in P_I \uplus T_I$ with $x <_K i_1'(i_1(z)) <_K x$.
  Then there is $a \in \{1,2\}$ and $y \in P_{K_a} \uplus T_{K_a}$ s.t. $i_a'(y) = x$. Since there are no images of interface elements in the causal relation between x and x, the causal relation is completely obtained from causal relations in K_a, i. e. we have $y <_{K_a} y$. This contradicts the fact that $<_{K_a}$ is irreflexive because K_a is an AHL-process net.

- **Case 2.** There is an element $z \in P_I \uplus T_I$ with $x <_K i'_1(i_1(z)) <_K x$.
  Due to the transitivity of $<_K$ there is $i'_1(i_1(z)) <_K i'_1(i_1(z))$ because

  $$i'_1(i_1(z)) <_K x <_K i'_1(i_1(z)).$$

  By statement (A) this implies $z <_{(i_1,i_2)} z$, contradicting the fact that by the composability of K_1 and K_2 w. r. t. (I, i_1, i_2) the induced causal relation $<_{(i_1,i_2)}$ is irreflexive.

Hence, $<_K$ is irreflexive.

Only If. Given the pushout diagram (PO) in the category **AHLPNets**. By Lemma 2 (PO) is also a pushout in the category **AHLNets**. We have to show that (K_1, K_2) are composable w. r. t. (I, i_1, i_2).

No cycles. Let $x, y \in P_I \uplus T_I$ with $x \prec_{(i_1,i_2)} y$.
Then by the definition of $\prec_{(i_1,i_2)}$ there is

$$i_1(x) <_{K_1} i_1(y) \quad \text{or} \quad i_2(x) <_{K_2} i_2(y)$$

and by the fact that $i'_1 \circ i_1 = i'_2 \circ i_2$, we have

$$i'_1 \circ i_1(x) <_K i'_1 \circ i_1(y)$$

because AHL-morphisms preserve pre and post conditions.
Since $<_K$ is transitive, we have also for the transitive closure $<_{(i_1,i_2)}$ of $\prec_{(i_1,i_2)}$, that $x <_{(i_1,i_2)} y$ implies $i'_1 \circ i_1(x) <_K i'_1 \circ i_1(y)$.

Now, let us assume that $<_K$ is not irreflexive, i. e. there is $x \in P_I \uplus T_I$ with $x <_{(i_1,i_2)} x$. Then there is

$$i'_1 \circ i_1(x) <_K i'_1 \circ i_1(x)$$

contradicting the fact that $<_K$ is irreflexive. Hence, $<_{(i_1,i_2)}$ is irreflexive.

Non-injective gluing. We have to show that for all $p_1 \neq p_2 \in IN(I)$ with $i_2(p_1) = i_2(p_2)$ there is $i_1(p_1) \in IN(K_1)$ or $i_1(p_2) \in IN(K_1)$.

So let $p_1 \neq p_2 \in IN(I)$ with $i_2(p_1) = i_2(p_2)$ and let us assume that $i_1(p_1) \notin IN(K_1)$ and $i_1(p_2) \notin IN(K_2)$. This means that there are $t_1, t_2 \in T_{K_1}$ and terms $term_1, term_2 \in T_\Sigma(X)$ with $(term_1, i_1(p_1)) \leq post_{K_1}(t_1)$ and $(term_2, i_1(p_2)) \leq post_{K_1}(t_2)$.

From preservation of post conditions by AHL-morphisms it follows that $(term_1, i'_1(i_1(p_1))) \leq post_K(i'_1(t_1))$ and $(term_2, i'_1(i_1(p_2))) \leq post_K(i'_1(t_2))$. Moreover, by commutativity of pushout (PO) we have

$$i'_1(i_1(p_1)) = i'_2(i_2(p_1)) = i'_2(i_2(p_2)) = i'_1(i_1(p_2))$$

We distinguish the following two cases.

Case 1. There is $i'_1(t_1) = i'_1(t_2)$.
Then we have $(term_1, i'_1(i_1(p_1))) \oplus (term_2, i'_1(i_1(p_1))) \leq post_K(i'_1(t_1))$, contradicting unarity of AHL-process net K.

Case 2. There is $i'_1(t_1) \neq i'_1(t_2)$.
Then $(term_1, i'_1(i_1(p_1))) \leq post_K(i'_1(t_1))$ and $(term_1, i'_1(i_1(p_1))) \leq post_K(i'_1(t_2))$ means that AHL-process net K has a backward conflict, which is also a contradiction.

Thus, we have $i_1(p_1) \in IN(K_1)$ or $i_1(p_2) \in IN(K_1)$.

The fact that for all $p_1 \neq p_2 \in OUT(I)$ with $i_2(p_1) = i_2(p_2)$ there is $i_1(p_1) \in OUT(K_1)$ or $i_1(p_2) \in OUT(K_1)$ follows analogously because AHL-morphisms preserve pre as well as post conditions and AHL-process net K also does not have any forward conflicts.

No conflicts. We have to show that $\forall x \in IN(I) : i_1(x) \notin IN(K_1) \Rightarrow i_2(x) \in IN(K_2)$. Let $x \in IN(I)$ with $i_1(x) \notin IN(K_1)$ and let us assume that there is $i_2(x) \notin IN(K_2)$.

Then $i_1(x)$ and $i_2(x)$ both are in the post domain of transitions, i. e. there are $t_1 \in T_{K_1}$ and $t_2 \in T_{K_2}$ such that $i_1(x) \in t_1\bullet$ and $i_2(x) \in t_2\bullet$. Since AHL-morphisms preserve post conditions there is

$$i'_1(i_1(x)) \in i'_1(t_1)\bullet \quad \text{and} \quad i'_2(i_2(x)) \in i'_1(t_2)\bullet$$

and due to the fact that (PO) commutes there is $i'_1(i_1(x)) = i'_2(i_2(x))$ which implies

$$i'_1(i_1(x)) \in i'_1(t_1)\bullet \cap i'_2(t_2)\bullet .$$

Since K is an AHL-process net it has no backward conflict implying that $i'_1(t_1) = i'_2(t_2)$. So due to the pushout property there is $t_0 \in T_I$ with

$$i_1(t_0) = t_1 \quad \text{and} \quad i_2(t_0) = t_2$$

Then by the fact that $i_1(x) \in i_1(t_0)\bullet$ together with the fact that i_1 is an AHL-morphism which preserves post domains it follows that $x \in t_0\bullet$. This contradicts the fact that $x \in IN(I)$. Hence, there is $i_2(x) \in IN(K_2)$.

Now, we show that $\forall x \in OUT(I) : i_1(x) \notin OUT(K_1) \Rightarrow i_2(x) \in OUT(K_2)$. Let $x \in OUT(I)$ with $i_1(x) \notin OUT(K_1)$ and let us assume that there is $i_2(x) \notin OUT(K_2)$.

Then $i_1(x)$ and $i_2(x)$ both are in the pre domain of transitions, i. e. there are $t_1 \in T_{K_1}$ and $t_2 \in T_{K_2}$ such that $i_1(x) \in \bullet t_1$ and $i_2(x) \in \bullet t_2$. Since AHL-morphisms preserve pre conditions there is

$$i'_1(i_1(x)) \in \bullet i'_1(t_1) \quad \text{and} \quad i'_2(i_2(x)) \in \bullet i'_1(t_2)$$

and by commutativity of (PO) we have $i'_1(i_1(x)) = i'_2(i_2(x))$ which implies

$$i'_1(i_1(x)) \in \bullet i'_1(t_1) \cap \bullet i'_2(t_2)$$

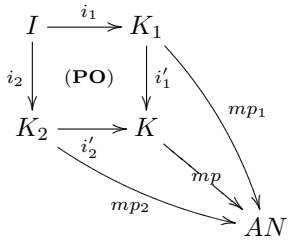
Since K is an AHL-process net it has no forward conflict implying that $i'_1(t_1) = i'_2(t_2)$. So due to the pushout property there is $t_0 \in T_I$ with

$$i_1(t_0) = t_1 \text{ and } i_2(t_0) = t_2$$

Then by the fact that $i_1(x) \in \bullet i_1(t_0)$ together with the fact that i_1 is an AHL-morphism which preserves pre domains it follows that $x \in \bullet t_0$. This contradicts the fact that $x \in OUT(I)$. Hence, there is $i_2(x) \in OUT(K_2)$.

Extension to Processes.

Given the pushout (PO) and additional AHL-morphisms $mp_1 : K_1 \to AN$ and $mp_2 : K_2 \to AN$ with $mp_1 \circ i_1 = mp_2 \circ i_2$.

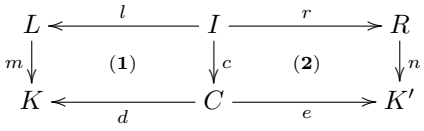


Then we also have a morphism $mp_0 : I \to AN$ defined by $mp_0 := mp_1 \circ i_1 = mp_2 \circ i_2$. Moreover the pushout property of (PO) implies a unique morphism $mp : K \to AN$ such that (PO) is also a pushout in the slice category $\mathbf{AHLNets} \setminus AN$. As shown above the composability of K_1 and K_2 w. r. t. (I, i_1, i_2) implies that K is an AHL-process net. Hence, $mp : K \to AN$ is an AHL-process which implies that (PO) is also pushout in the full subcategory $\mathbf{Proc(AN)} \subseteq \mathbf{AHLNets} \setminus AN$ of AHL-processes. □

A.4 Proof of Theorem 1 (Direct Transformation of AHL-Process Nets)

Given a production for AHL-process nets $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ and an AHL-process net K together with a morphism $m : L \to K$. Then the direct transformation of AHL-process nets with pushouts (1) and (2) in **AHLPNets** exists iff ϱ satisfies the transformation condition for AHL-process nets under m.

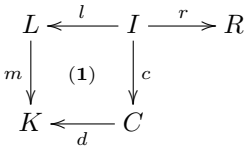
Extension to processes. In order to extend this construction for AHL-processes in the category **Proc**(**AN**) one additionally requires AHL-morphisms $mp : K \to AN$ and $rp : R \to AN$ with $mp \circ m \circ l = rp \circ r$. Then by composition of AHL-morphisms we obtain an AHL-process $cp = mp \circ d : C \to AN$ and the pushout (1) in **AHLPNets** is also a pushout of $mp \circ m$ and cp in **Proc**(**AN**). Moreover, the pushout (2) in **AHLPNets** provides a unique morphism $mp' : K' \to AN$ such that mp' is pushout of cp and rp in **Proc**(**AN**) according to Fact 4.



Proof. First, we prove the following lemma which states the equivalence of the gluing relation for a given production and match and the induced causal relation of the right-hand side of the production and the context net in **AHLNets**.

Lemma 3 (Gluing Relation Lemma)**.** *Given a production for AHL-process nets $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$, a match $m : L \to K$ where K is an AHL-process net, and pushout (1) in* **AHLNets***.*

Then the gluing relation $<_{(\varrho,m)}$ is exactly the induced causal relation of C and R w. r. t. (I, c, r), i. e. $<_{(\varrho,m)} \;=\; <_{(c,r)}$.



Proof. We define a relation $\prec_C \subseteq (P_C \times T_C) \uplus (T_C \times P_C)$ as follows:

$$\prec_C = \{(p,t) \in P_C \times T_C \mid p \in {}^\bullet t\} \cup \{(t,p) \in T_C \times P_C \mid p \in t^\bullet\}$$

The relation $\prec_C$ describes the direct causal relationship of the elements in C, i. e. the causal relation $<_C$ is the transitive closure of $\prec_C$. We show that for the relation $\prec_{(K,m)}$ in Def. 14 we have $\prec_{(K,m)} = \prec_C$, by showing that there is a subset relation in both directions.

Direction 1 ($\prec_{(K,m)} \subseteq \prec_C$). Let $x, y \in P_K \uplus (T_K \setminus m_T(T_L))$ with $x \prec_{(K,m)} y$. Due to the bipartite structure of Petri nets there are two possible cases:

- **Case 1.** There is $x \in P_K$ and $y \in T_K \setminus m_T(T_L)$.
  Due to the construction of C there is $y \in T_C$. Furthermore there is $term \in T_{OP}(X)_{type_K(x)}$ such that

$$\begin{aligned}(term, x) \leq pre_K(y) \quad &\Leftrightarrow \quad (term, x) \leq pre_K|_{T_C}(y) \\ &\Leftrightarrow \quad (term, x) \leq pre_C(y)\end{aligned}$$

  and hence $x \prec_C y$.
- **Case 2.** There is $x \in T_K \setminus m_T(T_L)$ and $y \in P_K$.
  In this case we have $x \in T_C$ and there is $term \in T_{OP}(X)_{type_K(x)}$ such that

$$\begin{aligned}(term, y) \leq post_K(x) \quad &\Leftrightarrow \quad (term, y) \leq post_K|_{T_C}(x) \\ &\Leftrightarrow \quad (term, y) \leq post_C(x)\end{aligned}$$

  and hence $x \prec_C y$.

Direction 2 ($\prec_C \subseteq \prec_{(K,m)}$). Let $x, y \in P_C \uplus T_C$ with $x \prec_C y$. Again we distinguish the two possible cases:

Case 1. There is $x \in P_C$ and $y \in T_C$.
Then there is $term \in T_{OP}(X)_{type_C(x)}$ such that $(term, x) \leq pre_C(y)$. Since AHL-morphisms preserve pre conditions and d is an inclusion we have

$$\begin{aligned}(term, x) \leq pre_C(y) &\Leftrightarrow (term, x) \leq d^{\oplus} \circ pre_C(y)\\ &\Leftrightarrow (term, x) \leq pre_K(d(y))\\ &\Leftrightarrow (term, x) \leq pre_K(y)\end{aligned}$$

So the fact that $T_C = T_K \setminus m_T(T_L)$ implies $x \prec_{(K,m)} y$.

Case 2. There is $x \in T_C$ and $y \in P_C$.
Then there is $term \in T_{OP}(X)_{type_C(x)}$ such that $(term, x) \leq post_C(y)$. Since AHL-morphisms preserve not only pre but also post conditions we obtain analogously to Case 1 that $x \prec_{(K,m)} y$.

So we have that $\prec_{(K,m)} = \prec_C$ and since $<_{(K,m)}$ is the transitive closure of $\prec_{(K,m)}$ and $<_C$ is the transitive closure of $\prec_C$ it follows that $<_{(K,m)} = <_C$.
Furthermore we can use the inclusion d to obtain from the commutativity of (1) that

$$m \circ l(x) = d \circ c(x) = c(x).$$

So let $\prec_{(c,r)} \subseteq (P_I \times T_I) \uplus (T_I \times P_I)$ be the relation defined by

$$\prec_{(c,r)} \;=\; \{(x, y) \mid c(x) <_C c(y) \vee r(x) <_R r(y)\}$$

then we have

$$\begin{aligned}\prec_{(\varrho,m)} &= \{(x, y) \in (P_I \times T_I) \uplus (T_I \times P_I) \mid m \circ l(x) <_{(K,m)} m \circ l(y) \vee r(x) <_R r(y)\}\\ &= \{(x, y) \in (P_I \times T_I) \uplus (T_I \times P_I) \mid c(x) <_C c(y) \vee r(x) <_R r(y)\}\\ &= \{(x, y) \in (P_I \times T_I) \uplus (T_I \times P_I) \mid r(x) <_R r(y) \vee c(x) <_C c(y)\}\\ &= \prec_{(r,c)}\end{aligned}$$

and since $<_{(\varrho,m)}$ is the transitive closure of $\prec_{(\varrho,m)}$ and $<_{(r,c)}$ is the transitive closure of $\prec_{(r,c)}$ we have $<_{(\varrho,m)} \;=\; <_{(r,c)}$.

□

Now we show that the pushouts (1) and (2) below exist in **AHLPNets** if and only if the production p under m satisfies the transformation condition for AHL-process nets.

$$\begin{array}{ccccc} L & \xleftarrow{l} & I & \xrightarrow{r} & R \\ {\scriptstyle m}\downarrow & (1) & {\scriptstyle c}\downarrow & (2) & {\scriptstyle n}\downarrow \\ K & \xleftarrow{d} & C & \xrightarrow{e} & K' \end{array}$$

If. Given production $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ satisfying the transformation condition for AHL-processes under match m. Since this implies that ϱ satisfies the the gluing condition for AHL-nets by Theorem 3 there exist pushouts (1) and (2) in **AHLNets**. We have to show that (1) and (2) are also pushouts in the category **AHLPNets** of AHL-process nets.

Pushout (1). From AHL-process net K and AHL-morphism $d : C \to K$ it follows by Lemma 1 that also C is an AHL-process net. So we have that all objects and morphisms in pushout (1) are in the full subcategory **AHLPNets** $\subseteq$ **AHLNets** which means that (1) is also a pushout in **AHLPNets**.

Pushout (2). We have to show that (R, C) are composable w. r. t. (I, r, c) (see Def. 13).

No cycles. The fact that the gluing relation $<_{(\varrho,m)}$ of ϱ und m is a strict partial order implies that the induced causal relation $<_{(r,c)}$ is a strict partial order because by Lemma 3 there is $x <_{(\varrho,m)} y \Leftrightarrow x <_{(r,c)} y$.

Non-injective gluing. From pushout (1) in **AHLPNets** of morphisms l and c it follows by Fact 4 that (L, C) are composable w. r. t. (I, l, c).

Let $p_1 \neq p_2 \in IN(I)$ with $c(p_1) = c(p_2)$. Then we have

$$m \circ l(p_1) = d \circ c(p_1) = d \circ c(p_2) = m \circ l(p_2)$$

which by the fact that ϱ and m satisfy the transformation condition implies that $r(p_1) \in IN(R)$ or $r(p_2) \in IN(R)$.

Analogously, we obtain for $p_1 \neq p_2 \in OUT(I)$ with $c(p_1) = c(p_2)$ that there is $r(p_1) \in OUT(R)$ or $r(p_2) \in OUT(R)$.

No conflicts. Let $x \in IN(I)$ and $c(x) \notin IN(C)$ then by the composability of (L, C) w. r. t. (I, l, c) follows that $l(x) \in IN(L)$. The fact that $c(x) \notin IN(C)$ implies $t \in T_C$ with $c(x) \in t\bullet$ and $d \circ c(x) \in d(t)\bullet$ because AHL-morphisms preserve post conditions. Due to the commutativity of (1) there is $m \circ l(x) = d \circ c(x)$ which means that $m \circ l(x) \notin IN(K)$ because $m \circ l(x) \in d(t)\bullet$. So there is $x \in InP$ and the fact that production ϱ and match m satisfy the transformation condition for AHL-processes implies that $r(x) \in IN(R)$.

Now, let $x \in OUT(I)$ and $c(x) \notin OUT(C)$ then by the composability of (L, C) w. r. t. (I, l, c) follows that $l(x) \in OUT(L)$. The fact that $c(x) \notin OUT(C)$ implies $t \in T_C$ with $c(x) \in \bullet t$ and $d \circ c(x) \in \bullet d(t)$ because AHL-morphisms preserve pre conditions. Due to the commutativity of (1) there is $m \circ l(x) = d \circ c(x)$ which means that $m \circ l(x) \notin OUT(K)$ because $m \circ l(x) \in \bullet d(t)$. So there is $x \in OutP$ and the fact that production ϱ and match m satisfy the transformation condition for AHL-processes implies that $r(x) \in OUT(R)$.

Thus, (R, C) are composable w. r. t. (I, r, c) leading to the existence of pushout (2) in **AHLPNets**.

Only If. Given pushouts (1) and (2) in **AHLPNets**. We have to show that the transformation condition for AHL-process nets (see Def. 15) is satisfied by production ϱ under match m.

Gluing condition. By Lemma 2 pushouts (1) and (2) in **AHLPNets** are also pushouts in **AHLNets** which by Fact 3 implies that the gluing condition is satisfied.

No cycles. By Fact 4 pushout (2) in **AHLPNets** implies that (R, C) are composable w. r. t. (I, r, c) which means that $<_{(r,c)}$ is a strict partial order. Due to Lemma 3 we know that there is $<_{(r,c)} = <_{(\varrho,m)}$ which means that also $<_{(\varrho,m)}$ is a strict partial order.

Non-injective gluing. Let $p_1 \neq p_2 \in IN(I)$ with $m \circ l(p_1) = m \circ l(p_2)$. Since l is injective, $p_1 \neq p_2$ implies $l(p_1) \neq l(p_2)$. Then due to the fact that (1) is a pushout, there is $p \in P_C$ with $c(p_1) = p = c(p_2)$. Thus, by composability of (R, C) w. r. t. (I, r, c) it follows that $r(p_1) \in IN(R)$ or $r(p_2) \in IN(R)$.

Analogously, we obtain for $p_1 \neq p_2 \in OUT(I)$ with $m \circ l(p_1) = m \circ l(p_2)$ that $r(p_1) \in OUT(R)$ or $r(p_2) \in OUT(R)$.

No conflicts. Let $x \in InP$ which means that $x \in IN(I)$ with $l(x) \in IN(L)$ and $m \circ l(x) \notin IN(K)$. The fact that $m \circ l(x) \notin IN(K)$ implies that there is $t \in T_K$ with $m \circ l(x) \in t\bullet$.

Let us assume that there is $t' \in T_L$ with $m_T(t') = t$. Then from the fact that m is an AHL-morphism, it follows that $l(x) \in t'\bullet$ because AHL-morphisms preserve post conditions. This contradicts the fact that $l(x) \in IN(K)$ and thus $t \notin m_T(T_L)$ which means that $t \in T_K \setminus m_T(T_L)$. Then by the construction of pushout complement T_C in **AHLNets** it follows that $t \in T_C$.

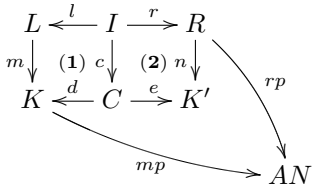
Moreover, we have $c(x) = m \circ l(x) \in t\bullet$ which means that $c(x) \notin IN(C)$. This implies that $r(x) \in IN(R)$ due to the composability of (R, C) w. r. t. (I, r, c) given by pushout (2) in **AHLPNets** and Fact 4.

Now, let $x \in OutP$ which means that $x \in OUT(I)$ with $l(x) \in OUT(L)$ and $m \circ l(x) \notin OUT(K)$. Then $m \circ l(x) \notin OUT(K)$ implies that there is $t \in T_K$ with $m \circ l(x) \in \bullet t$. Again, the assumption

that $t' \in T_L$ with $m_T(t') = t$ leads to a contradiction which means that $t \in T_K \setminus m_T(T_L)$. Then by the construction of T_C follows that $t \in T_C$ and we have $c(x) = m \circ l(x) \in {\bullet t}$ which means that $c(x) \notin OUT(C)$ and hence $r(x) \in OUT(R)$ by composability of (R, C) w. r. t. (I, r, c).

Extension to Processes.

Given pushouts (1) and (2) in **AHLPNets** and additional morphisms $mp : K \to AN$ and $rp : R \to AN$ with $mp \circ m \circ l = rp \circ r$.



Since L, C and I are AHL-process nets we obtain AHL-processes by composition of AHL-morphisms $lp := mp \circ m : L \to AN$, $cp := mp \circ d : C \to AN$ and $ip := mp \circ m \circ l = mp \circ d \circ c : I \to AN$ such that (1) is a commuting diagram in **Proc**(**AN**).

By Lemma 2 pushout (1) in **AHLPNets** is also a pushout in **AHLNets** and thus by construction of pushouts in slice categories it is also a pushout in **AHLNets** $\setminus AN$. Hence, due to the fact that lp, cp, ip and mp are AHL-processes we have that (1) is a pushout in the full subcategory **Proc**(**AN**) $\subseteq$ **AHLNets** $\setminus AN$.

Finally, we have

$$cp \circ c = mp \circ d \circ c = mp \circ m \circ l = rp \circ r$$

which by Fact 4 implies a unique morphism $mp' : K' \to AN$ such that (2) is also a pushout in **Proc**(**AN**). □

A.5 Proof of Theorem 2 (Extension of AHL-Process based on AHL-Net Transformation)

Given an AHL-net AN, an AHL-process $mp : K \to AN$ and a direct transformation $AN \overset{\varrho,m}{\Rightarrow} AN'$ with pushouts (1) and (2) in **AHLNets** as depicted in Figure 8. There exists an extension $mp' : K \to AN'$ of mp if and only if mp and ϱ, m satisfy the extension condition.

Proof. ***If.*** Let mp and p, m satisfy the extension condition. We define $mp_0 : K \to AN_0$ as $mp_0 = f^{-1} \circ mp$. Since f is injective, for the well-definedness of mp_0 it suffices to show that for all elements x in in K there exists an element y in AN_0 with $f(y) = mp(x)$. Let $p \in P_K$.

Case 1. There exists $x \in P_L$ with $m(x) = mp(p)$.
Then there is $x \in PP$ and since mp and ϱ, m satisfy the extension condition, we have $x' \in P_I$ with $l(x') = x$. Thus, we have $y = k(x')$ with $f(y) = f(k(x')) = m(l(x')) = m(x) = mp(p)$.

Case 2. There exists no $x \in P_L$ with $m(x) = mp(p)$.
Then by construction of pushout complements in **AHLNets** there exists $y \in AN_0$ with $f(y) = mp(p)$.

The proof for the existence of the transitions works analogously. Injectivitiy of f implies that we have a well-defined morphism $mp_0 : K \to AN_0$ with $f \circ mp_0 = f \circ f^{-1} \circ mp = mp$, and we obtain the required extension $mp' : K \to AN'$ by composition $mp' = g \circ mp_0$.

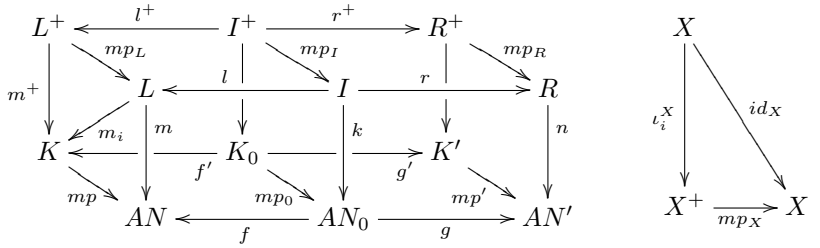
Only If. Let $mp' : K \to AN'$ be the extension of mp, i. e. there is $mp_0 : K \to AN_0$ with $f \circ mp_0 = mp$ and $g \circ mp_0 = mp'$. We have to show that all process points are gluing points. So, let $x \in PP$ and let w. l. o. g. $x \in P_L$. Then there is $p \in P_K$ with $mp(p) = m(x)$. Moreover, we have $y = mp_0(p) \in P_{AN_0}$ with $f(y) = f(mp_0(p)) = mp(p) = m(x)$. Since (1) is pushout in **AHLNets**, this implies that there is $x_0 \in P_I$ with $k(x_0) = y$ and $l(x_0) = x$. Hence, we have $x \in GP$.

□

A.6 Proof of Theorem 3 (Process Evolution based on Action Evolution)

Given an action evolution $AN \overset{\varrho,m}{\Longrightarrow} AN'$ via production $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$, and a process $mp : K \to AN$. Then there exists a production (ϱ^+, ϱ) for AHL-processes and a direct transformation $mp \overset{(\varrho^+,\varrho)}{\Longrightarrow} mp'$ as depicted in Figure 10a that realizes the changes described by ϱ on all occurrences in the process mp.
Construction: Let $(m_i : L \to K)_{i\in\mathcal{I}}$ be the class of all matches $m_i : L \to K$ with $mp \circ m_i = m$.

1. The production for AHL-process nets $\varrho^+ : L^+ \overset{l^+}{\leftarrow} I^+ \overset{r^+}{\rightarrow} R^+$ is defined as componentwise coproduct in **AHLPNets**:
   - $X^+ = \coprod_{i\in\mathcal{I}} X$ with injections $\iota_i^X : X \to X^+$ for $X \in \{L, I, R\}$,
   - $x^+ = \coprod_{i\in\mathcal{I}} x$ for $x \in \{l, r\}$
2. The processes $mp_X : X^+ \to X$ for $X \in \{L, I, R\}$ are the unique induced morphisms with $mp_X \circ \iota_i^X = id_X$ for all $i \in \mathcal{I}$ (see Figure 10b).
3. The match $m^+ : L^+ \to K$ is the unique induced morphism with $m^+ \circ \iota_i^L = m_i$ for all $i \in \mathcal{I}$.
4. K_0 and K' are constructed as direct AHL-process net transformation in the back of Figure 10a.
5. $mp_0 : K_0 \to AN_0$ is defined as $mp_0 = f^{-1} \circ mp \circ f'$, and $mp' : K' \to AN'$ is induced by the right pushout in the back of Figure 10a.



(a) Process Evolution

(b) Induced Process

Figure 12: Process evolution based on action evolution

Proof. We have to show that the construction given above is well-defined.

1. Since by Lemma 4 below the category **AHLPNets** has coproducts, we can construct the coproducts $X^+ = \coprod_{i\in\mathcal{I}} X$ for $X \in \{L, I, R\}$ and $x^+ = \coprod_{i\in\mathcal{I}} x$ for $x \in \{l, r\}$ in **AHLPNets**, and obtain coproduct injections $\iota_i^X : X \to X^+$ for $X \in \{L, I, R\}$ and $i \in \mathcal{I}$. Then ϱ^+ is a production for AHL-process nets because L^+, I^+ and R^+ are AHL-process nets.
2. The processes $mp_X : X^+ \to X$ for $X \in \{L, I, R\}$ are obtained as unique morphisms with $mp_X \circ \iota_i^X = id_X$ for all $i \in \mathcal{I}$ induced by the universal property of coproducts L^+, I^+ and R^+. Note that mp_X are retractions which in the case of AHL-morphisms means that $mp_{X,P}$ and $mp_{X,T}$ are surjective.

   It remains to show that (ϱ^+, ϱ) is a production for AHL-processes, i. e. that the top faces in Fig. 12a commute. By morphism $l : I \to L$ and coproduct I^+ of $(I)_{i\in\mathcal{I}}$ there exists a unique morphism $u : I^+ \to L$ such that for all $i \in \mathcal{I}$ there is $u \circ \iota_i^I = l$. So from $l \circ mp_I \circ \iota_i^I = l \circ id_I = l$ (see Fig. 12b) it follows that $l \circ mp_I = u$ by uniqueness of u. Hence, by compatibility of l^+ and l and Fig. 12b we have

   $$mp_L \circ l^+ \circ \iota_i^I = mp_L \circ \iota_i^L \circ l = id_L \circ l = l$$

   By uniqueness of u this implies $mp_L \circ l^+ = u = l \circ mp_I$. The commutativity $mp_R \circ r^+ = r \circ mp_I$ can be shown analogously. Hence, the top faces in Fig. 12a commute.

3. The match $m^+ : L^+ \to K$ with $m^+ \circ \iota_i^L = m_i$ for all $i \in \mathcal{I}$ is uniquely induced by coproduct L^+. Moreover, (m^+, m) is a **AHLProcs**-morphism because $mp \circ m^+ = mp \circ m_i \circ mp_L = m \circ mp_L$.

4. The existence of the direct transformation of AHL-process nets with pushouts (1) and (2) is shown in Lemma 5 below.

$$\begin{array}{ccccc} L^+ & \xleftarrow{l^+} & I^+ & \xrightarrow{r^+} & R^+ \\ {\scriptstyle m^+}\downarrow & (1) & \downarrow{\scriptstyle k^+} & (2) & \downarrow{\scriptstyle n^+} \\ K & \xleftarrow[f']{} & K_0 & \xrightarrow[g']{} & K' \end{array}$$

5. For the well-definedness of mp_0 as $mp_0 = f^{-1} \circ mp \circ f'$ we have to show that for all elements $x \in K_0$ there is a unique $y \in AN_0$ with $mp \circ f'(x) = f(y)$. Since ϱ is a production for action evolution, there is $P_L = l_P(P_I)$ which means that l_P is a bijection, i. e. an isomorphism in **Sets**. This implies that also l_P^+ is a bijection, and by the pushouts in the left front and back of the cube in Fig. 12a also f_P and f'_P are bijections. Thus, $mp_{0,P} = f_P^{-1} \circ mp_P \circ f'_P$ is well-defined. It remains to show that also $mp_{0,T}$ is well-defined. Let $t \in T_{K_0}$. In order to show the existence of $y \in T_{AN_0}$ with $mp \circ f'(t) = f(y)$, we distinguish the following two cases:

   Case 1. There is $f'(t) \in m^+(T_{L^+})$.
   This means that there is $t' \in T_{L^+}$ with $f'(t) = m^+(t')$. By pushout (1) we obtain $t_0 \in T_{I^+}$ with $l^+(t_0) = t'$ and $k^+(t_0) = t$. Then for $k(mp_I(t_0))$ we have

$$\begin{aligned} f(k(mp_I(t_0))) &= m(l(mp_I(t_0))) = m(mp_L(l^+(t_0))) \\ &= mp(m^+(l^+(t_0))) = mp(m^+(t')) \\ &= mp(f'(t)) \end{aligned}$$

   Case 2. There is $f'(t) \notin m^+(T_{L^+})$.
   Let us assume that there is $t' \in T_L$ and $i \in \mathcal{I}$ such that $m_i(t') = f'(t)$. Then we have $m^+(\iota_i^L(t')) = m_i(t') = f'(t)$ which is a contradiction. Thus, there is also $f'(t) \notin m_i(T_L)$ for all $i \in \mathcal{I}$. Since $(m_i)_{i \in \mathcal{I}}$ are all matches $m_i : L \to K$ with $mp \circ m_i = m$ it follows that $m(t_\varrho) \neq mp(f'(t))$ which means that $mp(f'(t)) \notin m(T_L)$. So by the pushout in the left front of Fig. 12a there is $y \in T_{AN_0}$ with $f(y) = mp(f'(t))$.

   The required uniqueness follows from injectivity of f which is implied by injectivity of l and the pushout in the left front of Fig. 12a. Hence, mp_0 is well-defined and we have

$$f \circ mp_0 = f \circ f^{-1} \circ mp \circ f' = mp \circ f'$$

   and

$$\begin{aligned} mp_0 \circ k^+ &= f^{-1} \circ mp \circ f' \circ k^+ = f^{-1} \circ mp \circ m^+ \circ l^+ \\ &= f^{-1} \circ m \circ mp_L \circ l^+ = f^{-1} \circ m \circ l \circ mp_I \\ &= f^{-1} \circ f \circ k \circ mp_I = k \circ mp_I \end{aligned}$$

   which means that the left cube in Fig. 12a commutes.

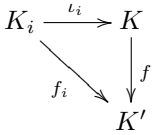
   Moreover, we have

$$\begin{aligned} g \circ mp_0 \circ k^+ &= g \circ k \circ mp_I = n \circ r \circ mp_I \\ &= n \circ mp_R \circ r^+ \end{aligned}$$

   which by pushout (2) implies a unique morphism $mp' : K' \to AN'$ such that the right cube in Fig. 12a commutes.

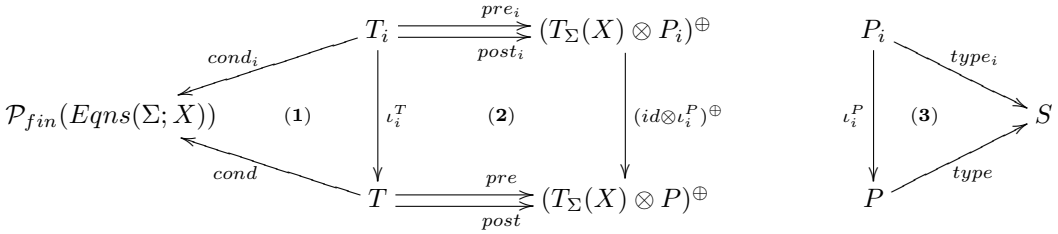
□

Lemma 4 (Coproduct of AHL-Process Nets)**.** *The categories* **AHLNets** *and* **AHLPNets** *have coproducts that can be constructed componentwise as disjoint unions in* **Sets***. This means, given AHL-(process) nets $(K_i)_{i\in\mathcal{I}}$, then there exists an AHL-(process) net K together with injections $\iota_i : K_i \to K$ such that K is the coproduct of $(K_i)_{i\in\mathcal{I}}$, written $K = \coprod_{i\in\mathcal{I}} K_i$, satisfying the following universal property: For all AHL-(process) nets K' and AHL-morphisms $(f_i : K_i \to K')_{i\in\mathcal{I}}$ there exists a unique morphism $f : K \to K'$ such that $f \circ \iota_i = f_i$ for all $i \in \mathcal{I}$.*



Proof. **Coproduct in AHLNets.** First, we show that **AHLNets** has coproducts.

Let $(K_i)_{i\in\mathcal{I}}$ be AHL-nets with $K_i = (\Sigma, P_i, T_i, pre_i, post_i, cond_i, type_i, A)$. We construct the coproducts $P = \coprod_{i\in\mathcal{I}} P_i$ with injections $(\iota_i^P : P_i \to P)_{i\in\mathcal{I}}$ and $T = \coprod_{i\in\mathcal{I}} T_i$ with injections $(\iota_i^T : T_i \to T)_{i\in\mathcal{I}}$ in **Sets**, given by the respective disjoint unions. Then, by the universal property of coproducts in **Sets** we obtain unique functions $cond : T \to \mathcal{P}_{fin}(Eqns(\Sigma; X))$, $pre, post : T \to (T_\Sigma(X) \otimes P)^\oplus$, and $type : P \to S$, such that diagrams (1), (2) and (3) below commute for all $i \in \mathcal{I}$.

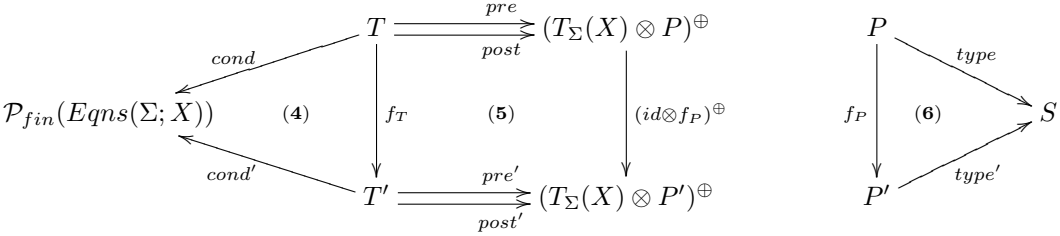


We define an AHL-net $K = (\Sigma, P, T, pre, post, cond, type, A)$ and AHL-morphisms $(\iota_i : K_i \to K)_{i\in\mathcal{I}}$ by $\iota_i = (\iota_i^P, \iota_i^T)$. The well-definedness follows from commutativity of diagrams (1)-(3) above.

Now, in order to show the universal property for K and $(\iota_i)_{i\in\mathcal{I}}$, let

$$K' = (\Sigma, P', T', pre', post', cond', type', A)$$

be an AHL-net and $(f_i : K_i \to K')_{i\in\mathcal{I}}$ AHL-morphisms. Using the universal property of P and T in **Sets**, we obtain unique functions $f_P : P \to P'$ and $f_T : T \to T'$, allowing us to define an AHL-morphism $f = (f_P, f_T) : K \to K'$. It remains to show that f is well-defined, i. e. we have to show that the diagrams (4)-(6) below commute.



Considering diagram (4) we have

$$cond' \circ f_T \circ \iota_i^T = cond' \circ f_{i,T} = cond_i$$

which by uniqueness of $cond$ implies that $cond' \circ f_T = cond$, i. e. diagram (4), and analogously diagram (6), commutes.

Let us now consider the pre-component of (5). By coproduct T of $(T_i)_{i\in\mathcal{I}}$ there is a unique morphism $g : T \to (T_\Sigma(X) \otimes P')^\oplus$ such that $g \circ \iota_i^T = (id \otimes f_P)^\oplus \circ (id \otimes \iota_i^P)^\oplus \circ pre_i$ for all $i \in \mathcal{I}$. Then, by commutativity of (2) we have for all $i \in \mathcal{I}$:

$$(id \otimes f_P)^\oplus \circ pre \circ \iota_i^T = (id \otimes f_P)^\oplus \circ (id \otimes \iota_i^P) \oplus \circ pre_i$$

implying that $g = (id \otimes f_P)^{\oplus} \circ pre$ by uniqueness of g. Moreover, we have for all $i \in \mathcal{I}$:

$$\begin{aligned} pre' \circ f_T \circ \iota_i^T &= pre' \circ f_{i,T} = (id \otimes f_{i,P})^{\oplus} \circ pre_i \\ &= (id \otimes (f_P \circ \iota_i^P))^{\oplus} \circ pre_i = (id \otimes f_P)^{\oplus} \circ (id \otimes \iota_i^P)^{\oplus} \circ pre_i \end{aligned}$$

which by uniqueness of g implies that $pre' \circ f_T = g = (id \otimes f_P)^{\oplus} \circ pre$. The proof for the *post*-component works analogously.

Hence, f is a well-defined AHL-morphism. The uniqueness of f follows from uniqueness of its components f_P and f_T.

Coproduct in AHLPNets. Let now $(K_i)_{i\in\mathcal{I}}$ be AHL-process nets. We show that the coproduct $K = \coprod_{i\in\mathcal{I}} K_i$ in **AHLNets** is also coproduct in **AHLPNets**. Analogously as in Lemma 2 it can be shown that coproducts in **AHLPNets** are also coproducts in **AHLNets**. Therefore, it suffices to show that the coproduct K in **AHLNets** is an AHL-process net. The conditions 1-4 in Def. 3 of AHL-process nets correspond only to connections between places and transitions. Since K consists only of disjoint copies of the AHL-process nets K_i which satisfy the conditions in Def. 3, also K satisfies these conditions, because there are no new connections between places and transitions which could violate the condition. Hence, K is an AHL-process net.

□

Lemma 5 (Process Net Evolution based on Action Evolution)**.** *Given an action evolution $AN \overset{\varrho,m}{\Longrightarrow} AN'$ via production $\varrho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$, and a process $mp : K \to AN$. Then the production $\varrho^+ : L^+ \overset{l^+}{\leftarrow} I^+ \overset{r^+}{\rightarrow} R^+$ and match m^+ as defined in Theorem 3 are applicable, i. e. the direct transformation of AHL-process nets with pushouts (1) and (2) below exist.*

$$\begin{array}{ccccc} L^+ & \xleftarrow{l^+} & I^+ & \xrightarrow{r^+} & R^+ \\ {\scriptstyle m^+}\downarrow & (1) & \downarrow{\scriptstyle k^+} & (2) & \downarrow{\scriptstyle n^+} \\ K & \xleftarrow[f']{} & K_0 & \xrightarrow[g']{} & K' \end{array}$$

Proof. For the existence of the direct transformation of AHL-process nets we have to show according to Theorem 1 that ϱ^+ and m^+ satisfy the transformation conditions 1-4 (see Def. 15):

1. *(Gluing Condition)* For satisfaction of the gluing condition we have to show that all identification and dangling points are gluing points (see Def. 10). Let us first consider the places, i. e. let $p \in (IP \cap P_{L^+}) \cup DP$. There is $mp_L(p) \in P_L$, and since ϱ is a production for action evolution, there is $p' \in P_I$ with $l(p') = mp_L(p)$. By construction of L^+ as coproduct, there is some $i \in \mathcal{I}$ such that $p = \iota_i^L(mp_L(p))$. Hence, we have $\iota_i^I(p') \in P_{I^+}$ with $l^+(\iota_i^I(p')) = \iota_i^L(l(p')) = \iota_i^L(mp_L(p)) = p$ which means $p \in GP$.

   Considering the transitions, let us assume $t \in IP \cap T_{L^+}$. Then there is $t' \in T_{L^+}$ with $t \neq t'$ and $m^+(t) = m^+(t')$. Since ϱ is a production for action evolution, we have $mp_L(t) = t_\varrho = mp_L(t')$ and there are $i, j \in \mathcal{I}$ with $\iota_i^L(t_\varrho) = t$ and $\iota_j^L(t_\varrho) = t'$. We show that $m_i = m_j$. First, we have

   $$m_i(t_\varrho) = m^+(\iota_i^L(t_\varrho)) = m^+(t) = m^+(t') = m^+(\iota_j^L(t_\varrho)) = m_j(t_\varrho)$$

   which means that $m_{i,T} = m_{j,T}$.

   Now, let $p \in P_L$. By production ϱ for action evolution, we have $p \in {\bullet t_\varrho} \cup t_\varrho{\bullet}$. W. l. o. g. let $p \in {\bullet t_\varrho}$ which means that there is $term \in T_\Sigma(X)$ such that $(term, p) \leq pre_L(t_\varrho)$. Let us assume that $m_i(p) = p_i \neq p_j = m_j(p)$. Since AHL-morphisms preserve pre conditions, we have $(term, p_i) \oplus (term, p_j) \leq pre_K(m_i(t_\varrho))$ and, moreover, there is $p' \in P_L$ with $m_i(p') = p_j$ and $(term, p') \leq pre_L(t_\varrho)$. From $p_i \neq p_j$ it follows that $p' \neq p$ which means that we have $(term, p) \oplus (term, p') \leq pre_L(t_\varrho)$, contradicting the fact that ϱ is a production for action evolution. Thus, we also have $m_{i,P} = m_{j,P}$, implying that we have similar matches $m_i = m_j$ and therefore $i = j$. So we have $t = \iota_i^L(t_\varrho) = \iota_j^L(t_\varrho) = t'$, contradicting the assumption $t \neq t'$ by $t \in IP \cap T_{L^+}$. Hence, m_T^+ is injective which means that there are no transitions that are identification points.

2. *(No Cycles)* We have to show that the gluing relation $<_{(\varrho^+,m^+)}$ defined as transitive closure of

$$\{(x,y) \in (P_{I^+} \times T_{I^+}) \uplus (T_{I^+} \times P_{I^+}) \mid m^+ \circ l^+(x) <_{(K,m^+)} m^+ \circ l^+(y) \vee r^+(x) <_R r^+(y)\}$$

is irreflexive. In order to show this we first show in the following that for all $x, y \in P_{I^+} \uplus T_{I^+}$ with $x <_{(\varrho^+,m^+)} y$ it follows that $m^+ \circ l^+(x) <_K m^+ \circ l^+(y)$.

So let us first consider the relation $<_{(K,m^+)}$, defined as transitive closure of

$$\{(x,y) \in (P_K \times (T_K \setminus m_T^+(T_{L^+}))) \uplus ((T_K \setminus m_T^+(T_{L^+})) \times P_K) \mid x \in \bullet y\}$$

Clearly, $<_{(K,m^+)}$ is a subset of $<_K$, containing only those relations between elements in K which are not or not only related via an occurrence $m_i(t_\varrho)$ for some $i \in \mathcal{I}$. So for $p_1, p_2 \in P_{I^+}$ we have that $m^+ \circ l^+(p_1) <_{(K,m^+)} m^+ \circ l^+(p_2)$ implies $m^+ \circ l^+(p_1) <_K m^+ \circ l^+(p_2)$.

It remains to show that for $p_1, p_2 \in P_{I^+}$ with $r^+(p_1) <_{R^+} r^+(p_2)$ there is $m^+ \circ l^+(p_1) <_K m^+ \circ l^+(p_2)$. For this purpose we first show that for places $p_1, p_2 \in P_I$ with $r(p_1) <_R r(p_2)$ it follows that $l(p_1) <_L l(p_2)$.

So let $p_1, p_2 \in P_I$ with $r(p_1) <_R r(p_2)$. Then there has to be at least one transition with $r(p_1)$ in its pre domain as well as a transition with $r(p_2)$ in its post domain which means that we have $r(p_1) \notin OUT(R)$ and $r(p_2) \notin IN(R)$. By contraposition of item 4 in Def. 20 we obtain $l(p_1) \notin OUT(L)$ and $l(p_2) \notin IN(L)$, and due to the fact that all places in L are in the environment of one transition t_ϱ it follows that $l(p_1) <_L l(p_2)$.

Since I^+, R^+ and L^+ contain only disjoint copies of I, R and L, respectively, we also have that for $p_1, p_2 \in P_{I^+}$ with $r^+(p_1) <_{R^+} r^+(p_2)$ it follows that $l^+(p_1) <_{L^+} l^+(p_2)$.

Furthermore, by the fact that AHL-morphisms preserve pre and post conditions, for all $t \in T_{I^+}$ and $p \in P_{I^+}$ it holds that $r^+(p) \in \bullet r^+(t) \cup r^+(t)\bullet$ implies $p \in \bullet t \cup t\bullet$ which in turn implies $l^+(p) \in \bullet l^+(t) \cup l^+(t)\bullet$. Thus, by transitivity of $<_{R^+}$ and $<_{L^+}$ it follows for all $x, y \in P_{I^+} \uplus T_{I^+}$ that $r^+(x) <_{R^+} r^+(y)$ implies $l^+(x) <_{L^+} l^+(y)$. Moreover, due to the preservation of pre and post conditions by AHL-morphisms, we obtain $m^+ \circ l^+(x) <_K m^+ \circ l^+(y)$.

Now, since $x <_{(\varrho^+,m^+)} y$ implies $m^+ \circ l^+(x) <_{(K,m)} m^+ \circ l^+(y)$ or $r^+(x) <_{R^+} r^+(y)$, from the facts that we have shown it follows that $x <_{(\varrho^+,m^+)} y$ implies $m^+ \circ l^+(x) <_K m^+ \circ l^+(y)$.

So let us now assume $x \in P_{I^+} \uplus T_{I^+}$ with $x <_{(\varrho^+,m^+)} x$. Then we have $m^+ \circ l^+(x) <_K m^+ \circ l^+(x)$, contradicting the fact that $<_K$ is irreflexive. Hence, $<_{(\varrho^+,m^+)}$ is irreflexive which means that it is a strict partial order.

3. *(Non-Injective Gluing)* Let $p_1 \neq p_2 \in IN(I^+)$ with $m^+ \circ l^+(p_1) = m^+ \circ l^+(p_2)$. We have to show that $r^+(p_1) \in IN(R^+)$ or $r^+(p_2) \in IN(R^+)$. We do this by showing that the negation leads to a contradiction.

So let us assume that $r^+(p_1), r^+(p_2) \notin IN(R^+)$. Then there are transitions $t_1, t_2 \in T_{R^+}$ with $r^+(p_1) \in t_1\bullet$ and $r^+(p_2) \in t_2\bullet$, and it follows that $mp_R(r^+(p_1)) \in mp_R(t_1)\bullet$ and $mp_R(r^+(p_2)) \in mp_R(t_2)\bullet$ because AHL-morphisms preserve post conditions.

So, by $mp_R \circ r^+ = r \circ mp_I$ we have $r(mp_I(p_1)), r(mp_I(p_2)) \notin IN(R)$, and by contraposition of item 4 in Def. 20 we have $l(mp_I(p_1)), l(mp_I(p_2)) \notin IN(L)$, and using $mp_L \circ l^+ = l \circ mp_I$ (see Fig. 13) we obtain $mp_L(l^+(p_1)), mp_L(l^+(p_2)) \notin IN(L)$. Due to construction of L^+ as coproduct of L there are $i, j \in \mathcal{I}$ such that $l^+(p_1) = \iota_i^L(mp_L(l^+(p_1)))$ and $l^+(p_2) = \iota_j^L(mp_L(l^+(p_2)))$. So, using again the fact that AHL-morphisms preserve pre conditions, it follows that $l^+(p_1), l^+(p_2) \notin IN(L^+)$. We distinguish the following two cases:

Case 1. There is $t \in T_{L^+}$ with $l^+(p_1), l^+(p_2) \in t\bullet$.
This means that there are $term_1, term_2 \in T_\Sigma(X)$ with $(term_1, l^+(p_1)) \oplus (term_2, l^+(p_2)) \leq post_{L^+}(t)$ and since AHL-morphisms preserve pre and post conditions, we have

$$(term_1, m^+(l^+(p_1))) \oplus (term_2, m^+(l^+(p_2))) \leq post_K(m^+(t)),$$

contradicting the fact that AHL-process net K is unary.

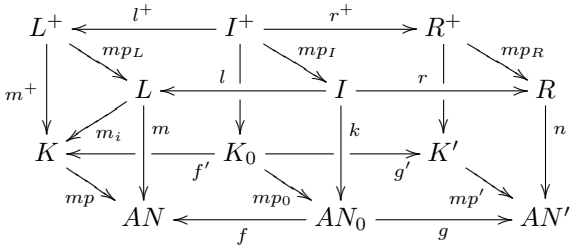


Figure 13: Process evolution based on action evolution

Case 2. There are $t_1 \neq t_2 \in T_{L^+}$ with $l^+(p_1) \in t_1\bullet$ and $l^+(p_2) \in t_2\bullet$.
Then we have $m^+(l^+(p_1)) = m^+(l^+(p_2)) \in m^+(t_1) \bullet \cap m^+(t_2)\bullet$ and, by injectivity of m_T^+ as shown item 1 (Gluing Condition) there is $m^+(t_1) \neq m^+(t_2)$, contradicting the fact that AHL-process net K does not have backward conflicts.

Hence, both cases lead to a contradiction, which means that there is $r^+(p_1) \in IN(R^+)$ or $r^+(p_2) \in IN(R^+)$.

The proof for $p_1 \neq p_2 \in OUT(I^+)$ with $m^+ \circ l^+(p_1) = m^+ \circ l^+(p_2)$ that $r^+(p_1) \in OUT(R^+)$ or $r^+(p_2) \in OUT(R^+)$ works analogously.

4. *(No Conflicts)* Let $x \in InP$, we have to show that $r^+(x) \in IN(R^+)$. Let us assume that $mp_L(l^+(x)) \notin IN(L)$ in order to show a contradiction. Then there is $t \in T_L$ with $mp_L(l^+(x)) \in t\bullet$. By construction of L^+ as coproduct there is some $i \in \mathcal{I}$ such that $\iota_i^L(mp_L(l^+(x))) = l^+(x)$. Thus, by the fact that AHL-morphisms preserve pre conditions, we obtain that $l^+(x) \in \iota_i^L(t)\bullet$, contradicting the fact that $l^+(x) \in IN(L^+)$ by definition of InP.

   So we have $mp_L(l^+(x)) \in IN(L)$ and by $l \circ mp_I = mp_L \circ l^+$ we have $l \circ mp_I(x) \in IN(L)$. By item 4 of Def. 20 we obtain $r \circ mp_I(x) \in IN(R)$. By $r \circ mp_I = mp_R \circ r^+$ we have $mp_R \circ r^+(x) \in IN(R)$.

   Let us now assume that $r^+(x) \notin IN(R^+)$ in order to show a contradiction, i. e. that there is $t \in T_{R^+}$ with $r^+(x) \in t\bullet$. Then due to preservation of post conditions by AHL-morhpisms we have $mp_R(r^+(x)) \in mp_R(t)\bullet$, contradicting the fact that $mp_R \circ r^+(x) \in IN(R)$.

   Thus, we have $r^+(x) \in IN(R^+)$, and hence $r^+(InP) \subseteq IN(R^+)$. The proof for $r^+(OutP) \subseteq OUT(R^+)$ works analogously.

□

References

[Apa11a] Apache Software Foundation. *http://apache.org*, August 2011.

[Apa11b] Apache Wave. *http://incubator.apache.org/wave/*, August 2011.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in TCS. Springer, 2006.

[EG11] H. Ehrig and K. Gabriel. Transformation of algebraic high-level nets and amalgamation of processes with applications to communication platforms. *International Journal of Software and Informatics*, 5, Part1:207–229, 2011.

[EHP+02] H. Ehrig, K. Hoffmann, J. Padberg, P. Baldan, and R. Heckel. High-level net processes. In *Formal and Natural Computing*, volume 2300 of *LNCS*, pages 191–219. Springer, 2002.

[Ehr79] H. Ehrig. Introduction to the algebraic theory of graph grammars (a survey). In V. Claus, H. Ehrig, and G. Rozenberg, editors, *Graph Grammars and Their Application to Computer Science and Biology, Lecture Notes in Computer Science,* No. 73, pages 1–69. Springer, 1979.

[Ehr05] H. Ehrig. Behaviour and Instantiation of High-Level Petri Net Processes. *Fundamenta Informaticae*, 65(3):211–247, 2005.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1.* Springer, 1985.

[ER97] Hartmut Ehrig and Wolfgang Reisig. An Algebraic View on Petri Nets. *Bulletin of the EATCS*, 61:52–58, February 1997.

[Gab10] K. Gabriel. Algebraic high-level nets and processes applied to communication platforms. Technical Report 2010/14, Technische Universität Berlin, 2010.

[Goo11] Google. *http://google.com*, August 2011.

[GR83] U. Goltz and W. Reisig. The Non-sequential Behavior of Petri Nets. *Information and Control*, 57(2/3):125–147, 1983.

[HM10] Kathrin Hoffmann and Tony Modica. Formal modeling of communication platforms using reconfigurable algebraic high-level nets. *ECEASST*, 30:1–25, 2010.

[Jen91] K. Jensen. Coloured petri nets: A high-level language for system design and analysis. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *LNCS*, pages 342–416. Springer, 1991.

[MGE+10] Tony Modica, Karsten Gabriel, Hartmut Ehrig, Kathrin Hoffmann, Sarkaft Shareef, Claudia Ermel, Ulrike Golas, Frank Hermann, and Enrico Biermann. Low- and High-Level Petri Nets with Individual Tokens. Technical Report 2009/13, Technische Universität Berlin, 2010. `http://www.eecs.tu-berlin.de/menue/forschung/forschungsberichte/2009`.

[MM90] J. Meseguer and U. Montanari. Petri Nets Are Monoids. *Information and Computation*, 88(2):105–155, 1990.

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic high-level net transformation systems. *Mathematical Structures in Computer Science*, 80:217–259, 1995.

[Pet62] C.A. Petri. *Kommunikation mit Automaten.* PhD thesis, Institut für instrumentelle Mathematik, Universität Bonn, 1962.

[Rei85] W. Reisig. *Petrinetze, Eine Einführung.* Springer Verlag, Berlin, 1985.

[Rei90] W. Reisig. Petri nets and algebraic specifications. Technische Universität München, SFB-Bericht 342/1/90 B, March, 1990.

[Roz87] G. Rozenberg. Behaviour of Elementary Net Systems. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets*, volume 254 of *LNCS*, pages 60–94. Springer, 1987.

[Roz97] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol 1:* Foundations. World Scientific, Singapore, 1997.

[Yon10] Tsvetelina Yonova. Formal description and analysis of distributed online collaboration platforms. Bachelor thesis, Technische Universität Berlin, 2010.