Electronic Communications of the EASST
Volume 51 (2012)

Proceedings of the
5th International Workshop on Petri Nets,
Graph Transformation and other Concurrency Formalisms
(PNGT 2012)

Process Evolution based on Transformation of Algebraic High-Level Nets with Applications to Communication Platforms

Karsten Gabriel

12 pages

Guest Editors: Julia Padberg, Kathrin Hoffmann
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Process Evolution based on Transformation of Algebraic High-Level Nets with Applications to Communication Platforms

Karsten Gabriel

kgabriel@cs.tu-berlin.de, Technische Universität Berlin

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. 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. Our main result is a construction for the evolution of AHL-processes based on the evolution of the corresponding AHL-net. This result can be used to transform scenarios in a communication platform according to the evolution of possibly multiple actions of the platform.

Keywords: High-Level Nets, Processes, Transformation, Communication Platforms

1 Introduction

High-level nets based on low-level Petri nets [Roz87, Rei85] and data types in the programming language 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]. 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. High-level processes for algebraic high-level nets, called AHL-net processes, have been introduced in [EHP$^+$02, Ehr05], which are high-level net morphisms $p : K \to AN$ with AN being an AHL-net, based on a suitable concept of high-level occurrence nets K.

The main aim of this paper is to give a short 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 Section 2 we introduce a small case study 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. Our main result presented in Section 4 shows how AHL-net processes can be transformed based on a special kind of transformation for AHL-nets, corresponding to multiple action evolution (i. e. the evolution of a set of features) of platforms, in contrast to single action evolution in [GE12]. Finally, the conclusion in Section 5 includes a summary of the paper.

2 Case Study: Communication 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 as Google Wave. Google itself has stopped the development of Google Wave, but the development is continued by the Apache Software Foundation as Apache Wave[Apa12].

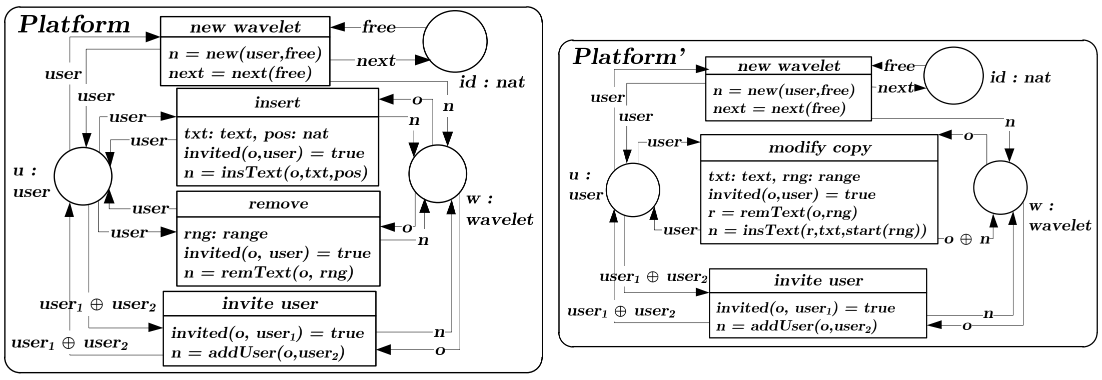
One of the most interesting aspects of Apache Wave is the possibility to make changes on previous contributions. Therefore, in contrast to email, text chat or forums, due to possible changes the resulting data of the communication does not necessarily give a comprehensive overview on all interactions 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.

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 modelling and analysis of Apache Wave very important in order to predict possible conflicts or other undesired behaviour 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 modelling technique of low-level Petri nets [Roz87, Rei85] and algebraic data types [EM85].

Figure 1a shows a small example of the structure of an AHL-net *Platform* which has three



(a) AHL-net *Platform* for an Apache Wave platform (b) Modified AHL-net *Platform*$'$

Figure 1: Apache Wave platforms

places and four 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*, *insert*, *remove* 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 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.

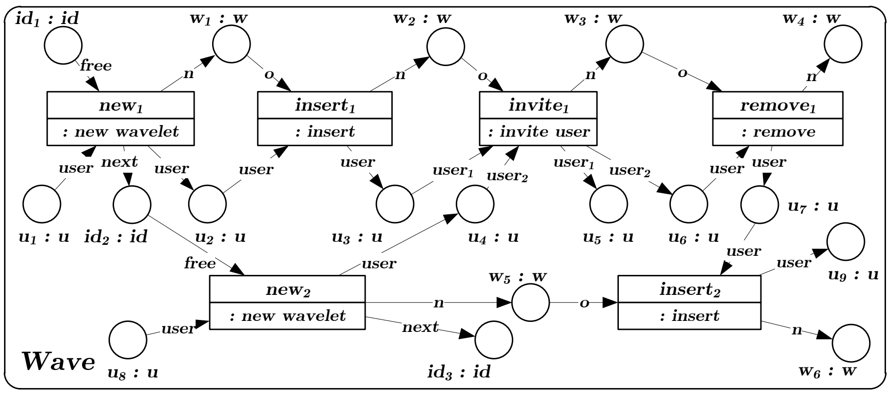


Figure 2: AHL-process *Wave* of a wave

As we have shown in [EG11] a suitable modelling technique for waves together with their histories are AHL-processes with instantiations. Figure 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 1b shows a net *Platform*$'$ which is an adaptation of our example *Platform* where the *insert* and *remove* transitions have been replaced by a *modify copy* transition which enables the user to replace text in a new copy of a wavelet while the original wavelet remains unchanged.

In order to model also the dynamic modification of scenario nets we need also rule-based modification of AHL-process nets. 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.

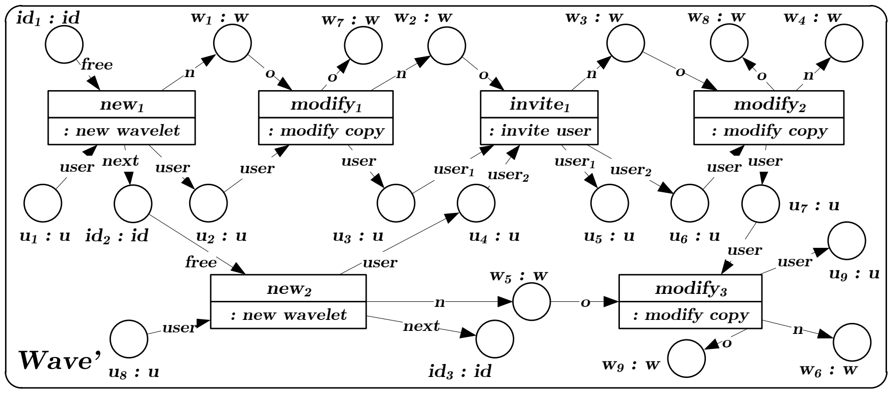


Figure 3: Modified AHL-process *Wave'*

An intuitive solution is to apply the modification of the platform also to the wave, replacing all occurrences of the old features with corresponding new ones if possible, or remove them otherwise. For this purpose, the rule which is used for modification of the platform should be equipped with additional information, describing how the changes of the platform affect occurrences of platform actions in a scenario. In the case of our example *Wave* and the platform evolution described above this leads to a new wave model *Wave'* depicted in Figure 3. The two occurrences of *insert* as well as the occurrence of *remove* have been replaced by occurrences of the transition *modify copy* and there are new *wavelet* places in the post domain of the new transitions, representing the unchanged originals. In our theorem in Section 4 we present a general construction to obtain under certain conditions a suitable modification of scenarios based on the evolution of the corresponding platform.

3 Modelling of Communication Platforms and Scenarios Using Algebraic High-Level 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 $s \oplus s' = \sum_{i=1}^{n} (\lambda_i + \lambda_i') x_i$.

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 $cond : T \to \mathscr{P}_{fin}(Eqns(\Sigma; X))$; a type of places $type : P \to S$ and a Σ-algebra A.

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, the restricted product $\otimes$ is defined by
$(T_{\Sigma}(X) \otimes P) = \{(term, p) | term \in T_{\Sigma}(X)_{type(p)}, p \in P\}$

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 $f = (f_\Sigma, f_P, f_T, f_A)$ with signature morphism f_Σ, functions f_P and f_T, and generalized algebra homomorphism f_A satisfying the following conditions:
(1) $(f_\Sigma^\# \otimes f_P)^\oplus \circ pre_1 = pre_2 \circ f_T$ and $(f_\Sigma^\# \otimes f_P)^\oplus \circ post_1 = post_2 \circ f_T$,
(2) $cond_2 \circ f_T = \mathscr{P}_{fin}(f_\Sigma^\#) \circ cond_1$, and
(3) $type_2 \circ f_P = f_\Sigma \circ type_1$.

The category defined by AHL-nets and AHL-net morphisms is denoted by **AHLNets** where the composition of AHL-net morphisms is defined componentwise. An example of an Apache Wave platform is given by the AHL-net *Platform* in Figure 1a (for more details, signature and algebra see [Gab12]).

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 v 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 v. For an example of the firing behaviour we refer to our technical report [Gab12].

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. 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 causal 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 explicitly in this paper, and we refer to our technical report [Gab12] where all constructions and results for AHL-process nets are also shown for their instantiations. 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 1 (Algebraic High-Level Process Nets and Processes) 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 $\bullet t = \{p_1, \dots, p_n\}$ and similarly $t\bullet$ we have

1. (*Unarity*): For all $p \in P$ there exist no terms $term_1, term_2 \in T_\Sigma(X)_{type(p)}$ such that $(term_1, p) \oplus (term_2, p) \leq pre(t)$ or $(term_1, p) \oplus (term_2, p) \leq post(t)$,
2. (*No Conflicts*): $\bullet t \cap \bullet t' = \emptyset$ and $t\bullet \cap t'\bullet = \emptyset$ for all $t, t' \in T, t \neq t'$, and
3. (*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.

AHL-process nets together with AHL-net morphisms between AHL-process nets form the full subcategory **AHLPNets** $\subseteq$ **AHLNets**. Similar to low-level processes, an AHL-process mp of an AHL-net AN is defined as an AHL-morphism $mp : K \to AN$ from an AHL-process net K into the net AN. The category **AHLProcs** of all AHL-processes is defined as full subcategory of the arrow category $\mathbf{AHLNets}^{\to}$ such that the objects are AHL-processes.

Example 1 (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 1a. 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 inserts some text into 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 removes something from the first wavelet before adding something to the second wavelet.*

4 Evolution of Communication Platforms and Scenarios

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] in the double pushout (DPO) approach.

A production (or transformation rule) for AHL-nets is a span $\rho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ of injective AHL-morphisms with isomorphic data type part, specifying a local modification of an AHL-net. It consists of a left-hand side L, an interface I which is the part of the left-hand side which is not deleted and a right-hand side R which additionally contains newly created net parts.

In order to add the new parts as specified in the right-hand side of a production to an AHL-net we use pushouts of AHL-nets as gluing construction. The diagram (1) in Figure 4 is a pushout (PO) diagram in the category **AHLNets** if (1) commutes and 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 is a unique $h : AN_3 \to AN_3'$ with $h \circ g_1 = h_1$ and $h \circ g_2 = h_2$.

$$\begin{tikzcd} AN_0 \arrow[r, "f_1"] \arrow[d, "f_2"'] \arrow[dr, phantom, "(1)"] & AN_1 \arrow[d, "g_1"] \\ AN_2 \arrow[r, "g_2"'] & AN_3 \end{tikzcd}$$

Figure 4: PO

We obtain the pushout by the componentwise quotient $(AN_1 \uplus AN_2)/_{\equiv}$ where $\equiv$ is the smallest relation with $f_1(x) \equiv f_2(x)$ for all x in each component of AN_0. Then in the resulting AHL-net AN_3 all elements that are matched by common interface elements are identified.

Let $\rho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ be a production and $m : L \to AN$ a (match) morphism. Then a *direct transformation* $AN \overset{(p,m)}{\Rightarrow} AN'$ in **AHLNets** is given by pushouts (2) and (3) shown in Figure 5. In [PER95] a gluing condition for AHL-nets is defined for the match $m : L \to AN$ and it is shown that the gluing condition is a necessary and sufficient condition for the existence of a direct transformation of AHL-nets.

$$\begin{tikzcd} L \arrow[d, "m"'] \arrow[dr, phantom, "(2)"] & I \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "c"] \arrow[dr, phantom, "(3)"] & R \arrow[d, "n"] \\ AN & C \arrow[l, "d"] \arrow[r, "e"'] & AN' \end{tikzcd}$$

Figure 5: DPO

Example 2 (Evolution of Apache Wave Platform) *In the bottom of Figure 6 is a production $\rho : L \overset{l}{\leftarrow} I \overset{r}{\rightarrow} R$ for AHL-nets that can be used for the evolution of an Apache Wave platform.*

The production describes a local modification that removes transitions insert and remove and inserts a new transition modify copy. Moreover, the newly created transition is connected to the former environment of the removed transitions. The production ρ can be applied to the AHL-net Platform in Figure 1a with a (match) morphism $m : L \to Platform$ which is an inclusion, leading to the modified AHL-net $Platform'$ in Figure 1b.

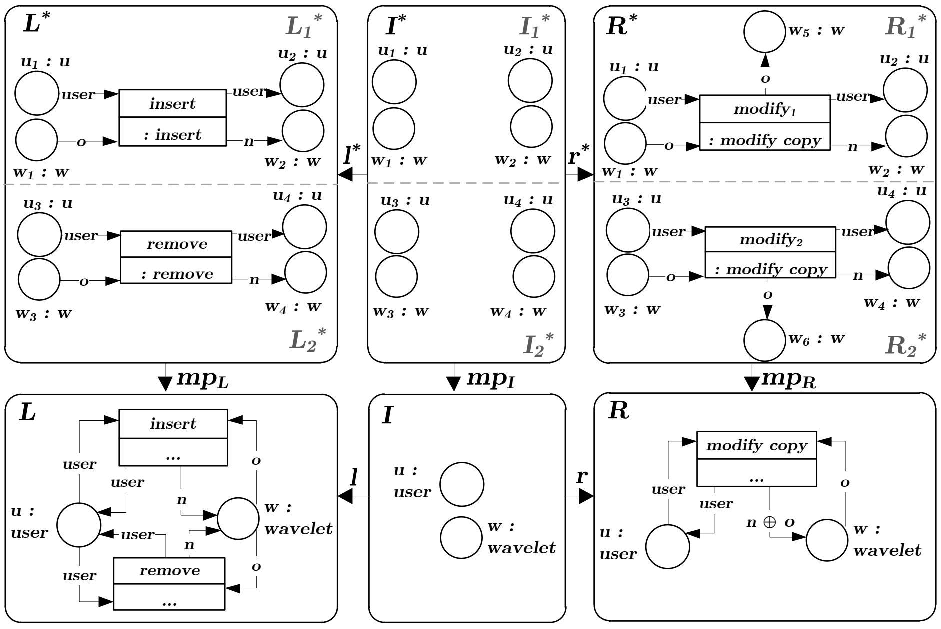


Figure 6: Production for action evolution

A production for AHL-process nets $\rho : L \xleftarrow{l} I \xrightarrow{r} R$ is defined as a span of injective **AHLPNets**-morphisms $l : I \to L$ and $r : I \to R$ with isomorphic data type part. 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 Definition 1. Accordingly, also the transformation of AHL-nets applied to an AHL-process net may introduce conflicts or cycles. Therefore, a *transformation condition* for AHL-process nets has been defined in [Gab10], and we have shown that the transformation condition is a necessary and sufficient condition that the direct transformation of an AHL-process net again leads to an AHL-process net. The satisfaction of the transformation condition by a production ρ and a match m requires that the gluing condition for AHL-nets is satsfied (see [PER95]). Moreover, it requires that the so-called gluing relation is irreflexive and that the application of the production does neither produce any conflicts nor violates the unarity condition of AHL-process nets. For details we refer to our technical report [Gab12], where we consider also the transformation of instantiations of AHL-processes. An example of the transformation of AHL-process nets is given at the end of this section.

Now consider again the platform evolution $Platform \Rightarrow Platform'$ from Example 2 and the scenario *wave* of *Platform* in Figure 2. There is no suitable morphism $wave' : Wave \to Platform'$

which means that the scenario *wave* is not a valid scenario of the modified platform. The reason is that the scenario *wave* contains occurrences of the actions *insert* and *remove*, but there is no corresponding action in the new platform *Platform*$'$. Nonetheless, the features to insert or remove some text in a wavelet has not been fully removed from the communication platform, but they have been replaced by the new action *modify copy* which does more or less the same as the old actions with the difference that insertion and deletion of text can be performed at once, and that it additionally produces a copy of the original wavelet. 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 $wave' : Wave' \to Platform'$ as depicted in Figure 3 where all occurrences of the actions *insert* and *remove* have been replaced by the new action *modify copy*.

In [GE12] we defined a construction for the transformation of scenarios based on (single) action evolutions of platforms. An important restriction of that construction was the requirement that the evolution of the platform can only change one single action. Since we replace two actions in our example, it is not possible to use that construction. Moreover, the scenario evolution based on single action evolution only allows to insert occurrences of actions in the scenario that have been newly introduced to the platform. Therefore, even an iterated application of that construction cannot be used for our example, since both of the actions should be replaced by the same action *modify copy* which would already exist after the first iteration.

In the following, we present a more general construction for the modification of scenarios based on the (multi) action evolution of platforms. For this purpose, since scenarios are modeled as AHL-processes, we need productions and the direct transformation of AHL-processes.

$$\begin{tikzcd}
K \arrow[r, Rightarrow, "\rho^*{,}m^*"] \arrow[d, "mp"'] \arrow[dr, phantom, "\xRightarrow{(\rho^*{,}\rho){,}(m^*{,}m)}"] & K' \arrow[d, "mp'"] \\
AN \arrow[r, Rightarrow, "\rho{,}m"'] & AN'
\end{tikzcd}$$

(a) Transformation of AHL-process

$$\begin{tikzcd}
L^* \arrow[dr, "mp_L"] \arrow[dd, "m^*"'] & & I^* \arrow[ll, "l^*"'] \arrow[rr, "r^*"] \arrow[dr, "mp_I"] \arrow[dd] & & R^* \arrow[dr, "mp_R"] \arrow[dd] & \\
& L \arrow[dd, "m"] & & I \arrow[ll, "l"'] \arrow[rr, "r"] \arrow[dd] & & R \arrow[dd] \\
K \arrow[dr, "mp"'] & & K_0 \arrow[ll] \arrow[rr] \arrow[dr, "mp_0"] & & K' \arrow[dr, "mp'"] & \\
& AN & & AN_0 \arrow[ll] \arrow[rr] & & AN'
\end{tikzcd}$$

(b) Commuting Cube

Figure 7: Transformation of AHL-process

A production for AHL-processes is a span $(\rho^*,\rho) : mp_L \xleftarrow{(l^*,l)} mp_I \xrightarrow{(r^*,r)} mp_R$ of injective **AHLProcs**-morphisms as shown in the top of Figure 7b. Let $(\rho^*,\rho) : mp_L \xleftarrow{(l^*,l)} mp_I \xrightarrow{(r^*,r)} mp_R$ be a production for AHL-processes and $(m^*,m) : mp_L \to mp$ a (match) morphism. Then a direct transformation $mp \overset{(\rho^*,\rho),(m^*,m)}{\Longrightarrow} mp'$ is given by the commuting cube in Figure 7b where the front and back faces are direct transformations of AHL-nets and AHL-process nets, respectively.

An action evolution is a direct transformation of AHL-nets that uses a special kind of production that is equipped with an *action evolution pattern*, containing information for each transition in the left-hand side, how an occurrence of that transition in a scenario has to be handled in order to update the scenario according to the changes of the AHL-net.

Definition 2 (Action Evolution) A production $\rho^* : L^* \xleftarrow{l^*} I^* \xrightarrow{r^*} R^*$ for AHL-process nets is called a *single action evolution pattern for AHL-processes* if it satisfies the following conditions:

1. *(Single Action)* L^* contains only one transition and its environment, i. e. $T_{L^*} = \{t_{\rho^*}\}$ and for all $p \in P_{L^*}$: $p \in \bullet t_{\rho^*} \cup t_{\rho^*}\bullet$,
2. *(Preserved Environment)* ρ^* is non-deleting on places, i. e. $P_{L^*} = l^*_P(P_{I^*})$, and
3. *(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 $l^*(p) \in OUT(L^*) \Rightarrow r^*(p) \in OUT(R^*)$, where $IN(K) = \{p \in P_K \mid \nexists t \in T_K : p \in t\bullet\}$ and $OUT(K) = \{p \in P_K \mid \nexists t \in T_K : p \in \bullet t\}$.

Moreover, a production ρ^* for AHL-process nets is called a *(multi) action evolution pattern for AHL-processes*, if it is a parallel production (i. e. the disjoint union) $\rho^* = \biguplus_{i\in\mathscr{I}} \rho^*_i$ of single action evolution patterns $(\rho^*_i)_{i\in\mathscr{I}}$.

Given in addition a production $\rho : L \xleftarrow{l} I \xrightarrow{r} R$ for AHL-nets, a production (ρ^*, ρ) for AHL-processes as shown in the top of Figure 7b is called a *production for action evolution* if

4. the transition-component $mp_{L,T}$ of the left-hand side is a bijection,
5. the left square is a pullback, and
6. the process net part ρ^* is an action evolution pattern for AHL-processes.

Given a production for action evolution (ρ^*, ρ), an AHL-net AN and a transition-injective match $m : L \to AN$ (i. e. m_T is injective), then a direct transformation $AN \overset{\rho,m}{\Longrightarrow} AN'$ is called *action evolution* with pattern ρ^*.

Example 3 (Action Evolution) *A production (ρ^*, ρ) for action evolution is shown in Figure 6 where the action evolution pattern ρ^* is the disjoint union of two single action evolution patterns ρ^*_1 and ρ^*_2, describing the replacement of occurrences of* insert *and* remove*, respectively, by occurrences of the new transition* modify copy*. Note that the left square in Figure 6 is a pullback, but the right square only commutes, since the right-hand side R^* contains occurrences w_5 and w_6 of the place w in R which is also contained in the interface I, but there are no corresponding places in the interface I^*.*

In order to determine how the action evolution pattern of a production for action evolution has to be used for the evolution of a corresponding AHL-process net, we have to make a choice of matches as defined in the following. Note, however, that in a “proper” high-level net with a sufficiently complex data type part, like our example in Figure 1, it is possible to determine the “role” of each place by using distinctive term inscriptions for different places in the pre respectively post domain of a transition. In that case there is only one possible choice of matches for the elements of an action evolution to their occurrences in a corresponding AHL-process net, and therefore, there is only one possible interpretation for the handling of these occurrences.

Definition 3 (Action Occurrences and Choices of Matches) Let $AN \overset{\rho,m}{\Longrightarrow} AN'$ be an action evolution with pattern $\rho^* = \biguplus_{i\in\mathscr{I}} \rho^*_i$, and a process $mp : K \to AN$. The family $occ = (occ_i)_{i\in\mathscr{I}}$ of all *action occurrences* is defined by $occ_i = \{t \in T_K \mid mp(t) = m \circ mp_L \circ \iota^L_i(t_{\rho_i})\}$ for all $i \in \mathscr{I}$, where ι^L_i is the injection $\iota^L_i : L^*_i \to L^*$ and t_{ρ_i} is the single transition in T_{L_i} (see item 1 in Definition 2).

An AHL-morphism $m_{i,o} : L^*_i \to K$ is called *match for occurrence* $o \in occ_i$ if $m_{i,o}(t_{\rho_i}) = o$ and $m_{i,o}$ is compatible with m and mp, i. e. $mp \circ m_{i,o} = m \circ mp_L \circ \iota^L_i$. A *choice of matches for* *occurrences* is a family $(m_{i,o} : L_i^* \to K)_{i \in \mathscr{I}, o \in occ_i}$ such that for every $i \in \mathscr{I}$ and $o \in occ_i$ we have that $m_{i,o}$ is a match for occurrence o.

Now, the following theorem states that for every process $mp : K \to AN$ and an action evolution $AN \Rightarrow AN'$ together with a choice of matches of occurrences there exists a corresponding transformation of AHL-processes $mp \Rightarrow mp'$ with $mp' : K' \to AN'$. 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. For a proof of the theorem, we refer to our technical report [Gab12].

Theorem (Process Evolution based on Action Evolution). *Let $AN \overset{\rho,m}{\Longrightarrow} AN'$ be an action evolution with pattern $\rho^* = \biguplus_{i \in \mathscr{I}} \rho_i^*$, and $mp : K \to AN$ an AHL-process. Then for every choice of matches for occurrences $(m_{i,o} : L_i^* \to K)_{i \in \mathscr{I}, o \in occ_i}$ there exists a production (ρ^+, ρ) for AHL-processes and a direct transformation $mp \overset{(\rho^+,\rho)}{\Longrightarrow} mp'$ as depicted in Figure 8. In this case, mp' is called process evolution of mp based on action evolution $AN \overset{\rho,m}{\Longrightarrow} AN'$ with pattern ρ^*.*

Construction and proof sketch.

1. The action evolution pattern ρ^+ is constructed as parallel production (disjoint union) $\rho^+ = L^+ \overset{l^+}{\leftarrow} I^+ \overset{r^+}{\to} R^+ = \biguplus_{i \in \mathscr{I}} \biguplus_{o \in occ_i} \rho_i^*$ with coproduct injections $\mu_{i,o}^X : X_i^* \to X^+$ for $X \in \{L, I, R\}$, which together with morphisms $mp_X \circ \iota_i^X$ induce unique morphisms $mp_X^+ : X^+ \to X$ such that $(\rho^+, \rho) = mp_L^+ \overset{(l^+,l)}{\leftarrow} mp_I^+ \overset{(r^+,r)}{\to} mp_R^+$ is a production for AHL-processes.
2. A match m^+ is induced by disjoint union $L^+ = \biguplus_{i \in \mathscr{I}} \biguplus_{o \in occ_i} L_i^*$ and matches $m_{i,o}$.
3. Then, since action evolution patterns are always applicable with a given transition-injective match as shown in [Gab12], we obtain a direct transformation $K \overset{\rho^+,m^+}{\Longrightarrow} K'$ of AHL-process nets in the lower back of Figure 8.
4. The process $mp_0 : K_0 \to AN_0$ can be obtained by construction of K_0 as pullback in the left bottom of Figure 8, and the process $mp' : K' \to AN'$ is induced by universal property of the pushout in the lower right back of the cube.
5. The direct transformation $mp \overset{(\rho^+,\rho)}{\Longrightarrow} mp'$ is given by the lower double cube in Figure 8. $\square$

$$\begin{tikzcd}
L_i^* \arrow[dr, "\iota_i^L"] \arrow[dd, dotted, "\mu_{i,o}^L"'] \arrow[dddd, bend right=60, "m_{i,o}"'] & & I_i^* \arrow[ll, "l_i^*"'] \arrow[rr, "r_i^*"] \arrow[dr, "\iota_i^I"] \arrow[dd, dotted] & & R_i^* \arrow[dr, "\iota_i^R"] \arrow[dd, dotted] & \\
& L^* \arrow[dd, "mp_L"] & & I^* \arrow[ll, "l^*"'] \arrow[rr, "r^*"] \arrow[dd, "mp_I"] & & R^* \arrow[dd, "mp_R"] \\
L^+ \arrow[dr, dotted, "mp_L^+"] \arrow[dd, dotted, "m^+"'] & & I^+ \arrow[ll, dotted, "l^+"'] \arrow[rr, dotted, "r^+"] \arrow[dr, dotted, "mp_I^+"] \arrow[dd, dotted] & & R^+ \arrow[dr, dotted, "mp_R^+"] \arrow[dd, dotted] & \\
& L \arrow[dd, "m"] & & I \arrow[ll, "l"'] \arrow[rr, "r"] \arrow[dd, "k"] & & R \arrow[dd, "n"] \\
K \arrow[dr, "mp"] & & K_0 \arrow[ll, dotted, "f'"'] \arrow[rr, dotted, "g'"] \arrow[dr, dotted, "mp_0"] & & K' \arrow[dr, dotted, "mp'"] & \\
& AN & & AN_0 \arrow[ll, "f"] \arrow[rr, "g"'] & & AN'
\end{tikzcd}$$

Figure 8: Process evolution based on action evolution

Example 4 (Evolution of Scenario based on Platform Evolution) *Consider again the action evolution $Platform \Rightarrow Platform'$ via production ρ in the bottom of Figure 6, and the scenario* *wave* : *Wave* $\rightarrow$ *Platform in Figure 2. There are two occurrences of the action* insert *and one occurrence of the action* remove*, and for each occurrence there is only one possible match for that occurrence. Thus, our construction yields a production ρ^+ containing two copies of ρ_1^* and one copy of ρ_2^* which are depicted in the top of Figure 6. Moreover, we obtain a match m^+ induced by our (unique) choice of matches, such that ρ^+ is applicable with match m^+ and we obtain a new scenario* $wave'$: $Wave' \rightarrow Platform'$ *depicted in Figure 3 where all occurrences of* insert *and* remove *actions have been replaced by corresponding occurrences of* modify copy *actions.*

5 Conclusion

In this paper we have shown that AHL-nets, AHL-processes, and AHL-transformations can be considered as an 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, Mod12] and Google Wave [EG11]. In this paper we have extended the general framework in order to model the evolution of Apache Wave platforms and scenarios, 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 explicitely in [Gab11].

Our main result shows how AHL-net processes can be transformed semi-automatically based on a special kind of transformation for AHL-nets, corresponding to the evolution of possibly multiple actions of platforms. In future work we will analyse conflicts between actions in communication platforms by analysis of process evolutions. Moreover, we will study under which conditions the analysis results for given scenarios can be transferred to scenarios of evolved platforms using the results presented in this paper. Finally, we are planning to develop a tool support for the modelling and analysis of AHL-nets and their processes.

Bibliography

[Apa12] Apache Wave. http://incubator.apache.org/wave/. September 2012.

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

[EG11] H. Ehrig, 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, R. Heckel. High-Level Net Processes. In *Formal and Natural Computing*. LNCS 2300, pp. 191–219. Springer, 2002.

[Ehr79] H. Ehrig. Introduction to the Algebraic Theory of Graph Grammars (A Survey). In Claus et al. (eds.), *Graph Grammars and Their Application to Computer Science and Biology, Lecture Notes in Computer Science,* No. 73. Pp. 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, B. Mahr. *Fundamentals of Algebraic Specification 1*. Springer, 1985.

[Gab10] K. Gabriel. Algebraic High-Level Nets and Processes Applied to Communication Platforms. Technical report 2010/14, Technische Universität Berlin, 2010.

[Gab11] K. Gabriel. Modelling Evolution of Communication Platforms and Scenarios based on Transformations of High-Level Nets and Processes – Extended Version. Technical report 2011/08, Technische Universität Berlin, 2011.

[Gab12] K. Gabriel. Process Evolution based on Transformation of Algebraic High-Level Nets with Applications to Communication Platforms – Extended Version. Technical report, Technische Universtät Berlin, 2012. To appear.

[GE12] K. Gabriel, H. Ehrig. Modelling evolution of communication platforms and scenarios based on transformations of high-level nets and processes. *Theor. Comput. Sci.* 429:87–97, Apr. 2012.

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

[HM10] K. Hoffmann, T. 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 Rozenberg (ed.), *Advances in Petri Nets 1990*. LNCS 483, pp. 342–416. Springer, 1991.

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

[Mod12] T. Modica. *Formal Modeling, Simulation, and Validation of Communication Platforms*. PhD thesis, Technische Universität Berlin, 2012.

[PER95] J. Padberg, H. Ehrig, L. Ribeiro. Algebraic High-Level Net Transformation Systems. *Mathematical Structures in Computer Science* 80:217–259, 1995.

[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*. LNCS 254, pp. 60–94. Springer, 1987.

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