Composition and Independence of High-Level Net Processes

Hartmut Ehrig, Kathrin Hoffmann*, Karsten Gabriel*, Julia Padberg

Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany

Abstract. Based on the notion of processes for low-level Petri nets we analyse in this paper high-level net processes defining the non-sequential behaviour of high-level nets. In contrast to taking low-level processes of the well known flattening construction for high-level nets our concept of high-level net processes preserves the high-level structure. The main results are the composition, equivalence and independence of high-level net processes under suitable conditions. Independence means that they can be composed in any order leading to equivalent high-level net processes which especially have the same input/output behaviour. All concepts and results are explained with a running example of the "House of Philosophers", a high-level net extension of the classical "Dining Philosophers".

Keywords: High-level net models, analysis of nets, behaviour of nets, high-level net processes, composition, equivalence, independence.

1 Introduction

For low-level Petri nets it is well known that processes are essential to capture their non-sequential truly concurrent behaviour (see e.g. [1–5]). Processes for high-level nets are often defined as processes of the low-level net which is obtained from flatting the high-level net. In [6, 7] we have defined high-level net processes for high-level nets based on a suitable notion of high-level occurrence nets which are defined independently of the flattening construction. The flattening of a high-level occurrence net is in general not a low-level occurrence net due to so called assignment conflicts in the high-level net.

The essential idea is to generalise the concept of occurrence nets from the low-level to the high-level case. This means that the net structure of a high-level occurrence net has similar properties like a low-level occurrence net, i.e. unitarity, conflict freeness, and acyclicity. But we drop the idea that an occurrence net captures essentially one concurrent computation. Instead, a high-level occurrence net and a high-level process are intended to capture a set of different concurrent computations corresponding to different input parameters of the process. In fact, high-level processes can be considered to have a set of initial markings for the

* This work has been partly funded by the research project forMA$_l$NET (see http://tfs.cs.tu-berlin.de/formalnet/) of the German Research Council.

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

In this paper we extend the notion of high-level net processes with initial markings by a set of corresponding instantiations. An instantiation is a subnet of the flattening defining one concurrent computation of the process. The advantage is that we fix for a given initial marking a complete firing sequence where each transition fires exactly once.

The main ideas and results in this paper concern the composition of high-level net processes. In general the composition of high-level net processes is not a high-level net process, because the composition may contain forward and/or backward conflicts as well as the partial order might be violated. Thus we state suitable conditions, so that the composition of high-level processes leads to a high-level process.

We introduce the concept of equivalence of high-level net processes, where the net structures of these high-level net processes might be different, but they have especially the same input/output behaviour. Hence their concurrent computations are compared in the sense that they start and end up with the same marking, but even corresponding dependent transitions may be fired in a different order. The main problem in this context which is solved in this paper is to analyse the independence of high-level net processes, i.e. under which condition high-level processes can be composed in any order leading to equivalent processes.

The paper is organised as follows. In Section 2 we explain the concepts and results of this paper using the "House of Philosophers" from [8] as an example. In Section 3 on the one hand we review the notions for high-level net processes and on the other hand we introduce the new notion of high-level net processes with instantiations. In Section 4 we present our main theorems concerning the composition, equivalence and independence of high-level net processes. In this section we give proof sketches and the detailed proofs can be found in the Appendix. Finally we conclude with related work and some interesting aspects of future work in Section 5.

2 House of Philosophers

In this section we review our example of the "House of Philosophers" [8] in order to illustrate the concepts in the following sections. This example is an extension of the well-known classical "Dining Philosophers" where in addition philosophers may move around e.g. by leaving and entering a table in the restaurant. For this reason three different locations, the library, the entrance-hall, and the restaurant, are represented by places in the algebraic high-level (AHL) net in Fig. 1.

The marking of the AHL-net shows the distribution of the philosophers at different places in the house. Initially there are two philosophers at the library, one philosopher at the entrance-hall, and four additional philosophers are at the two tables in the restaurant. The mobility aspect of the philosophers is modeled by transitions termed *enter* and *leave library* as well as *enter* and *leave restaurant* in Fig. 1, while the static structure of the net for philosophers is changed by rule-based transformations using the rules $rule_1, \ldots, rule_4$. The transitions *start/stop reading* and *start/stop activities* realise the well known token game.

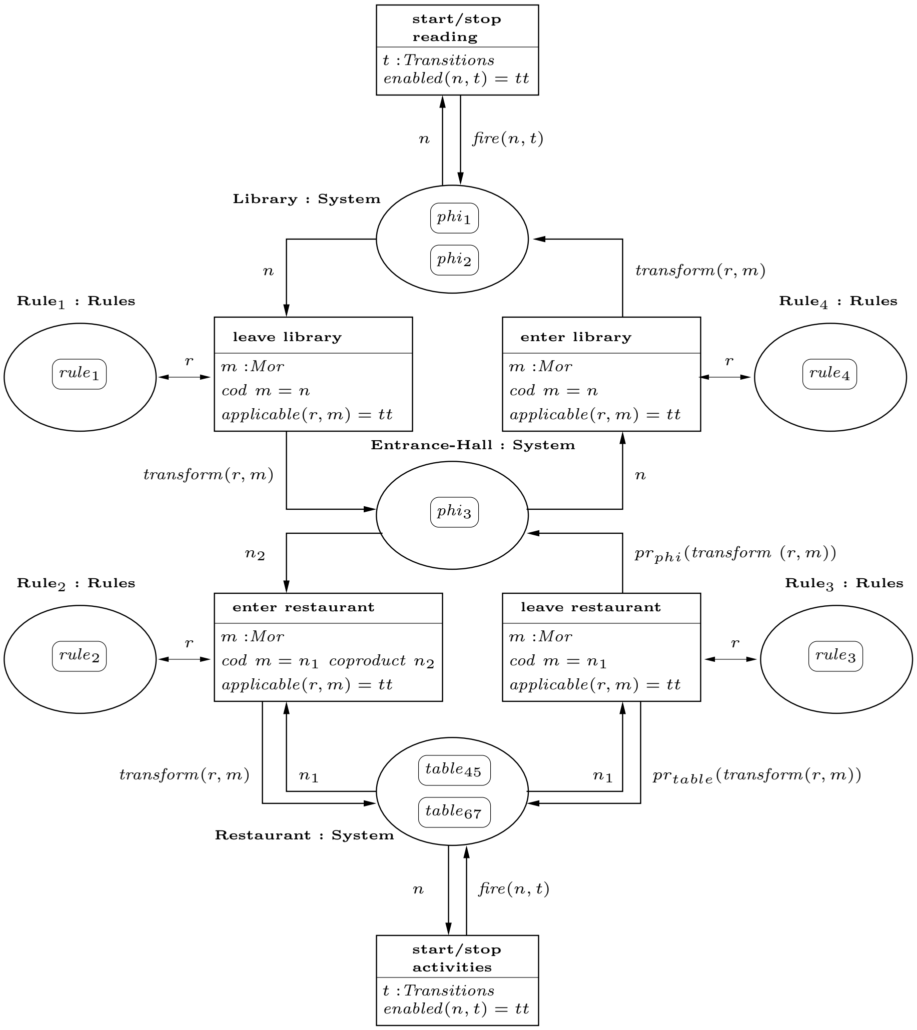

Fig. 1. AHL-net AN_{House} of the "House of Philosophers"

In the following we concentrate on the behaviour of the transitions *start/stop activities* and *enter restaurant*, while a detailed explanation of the other transitions and the corresponding formal framework can be found in Section 3, the Appendix A.1 and [8].

On the left hand side of Fig. 2 the P/T-system of the table table_{45} is depicted, where both the philosophers 4 and 5 are in the state *thinking*. The P/T-system is used as a token on the place *Restaurant* in Fig. 1. To start eating we use the transition *start/stop activities* of the AHL-net in Fig. 1. First we give an assignment of the variables v_1 and assign the table table_{45} to the variable n and the transition that realises the eating of philosopher 5 to the variable t. The firing condition checks that the philosopher 5 has his left and right forks. The evaluation of the net inscription $fire(n, t)$ realises the well-known token game by computing the follower marking of the P/T-system and we obtain the new P/T-system table'_{45} depicted on the right hand side of Fig. 2, where the philosopher 5 is eating.

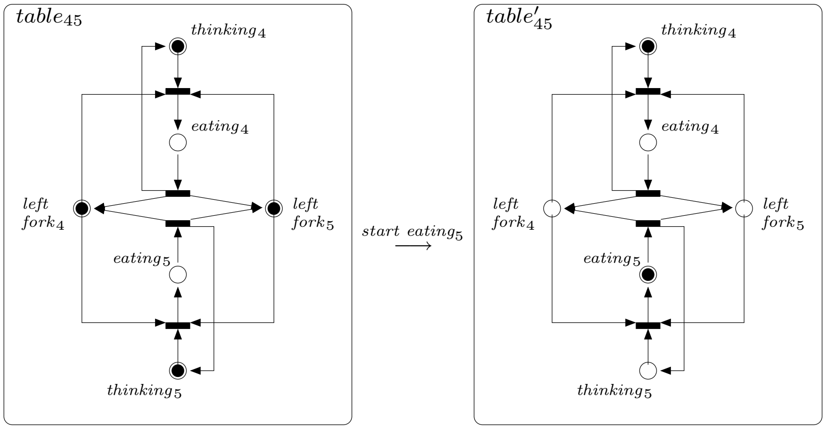

Fig. 2. P/T-systems table_{45} and table'_{45} of philosophers 4 and 5

Assume the philosopher 3, consisting of one marked place and one transition with corresponding arrows, would like to enter the restaurant in order to take place as a new guest at the table table_{45} (see left hand side of Fig. 2), so that the seating arrangement of the table has to be changed. Formally, we apply the rule rule_2, which is depicted in the upper row of Fig. 3 and used as token on place Rule_2. In general a rule $r = (L \overset{i_1}{\leftarrow} I \overset{i_2}{\rightarrow} R)$ is given by three P/T-systems called left-hand side, interface, and right-hand side respectively and the application of a rule discussed below describes the replacement of the left-hand side by the right-hand side preserving the interface.

The philosopher 3 sits down at table table_{45} by firing the transition *enter restaurant* in the AHL-net in Fig. 1 using the following assignment of the variables n_1, n_2, r and m given in the net inscriptions of the transition *enter restaurant*: $v_2(n_1) = \mathit{table}_{45}$, $v_2(n_2) = \mathit{phi}_3$, $v_2(r) = \mathit{rule}_2$, and $v_2(m) = g$ (see match morphism $g : L_2 \to (\mathit{phi}_3, \mathit{table}_{45})$ in Fig. 3). In our case the match g maps $\mathit{thinking}_j$ and eating_j in L_2 to $\mathit{thinking}_5$ and eating_5 in $(\mathit{phi}_3, \mathit{table}_{45})$. In the first step we compute the disjoint union of the P/T-system phi_3 and the P/T-system $table_{45}$ as denoted by the net inscription n_1 *coproduct* n_2 resulting in the P/T-system $(phi_3, table_{45})$ in Fig. 3. The firing conditions makes sure that on the one hand the rule is applied to the P/T-system $(phi_3, table_{45})$ and on the other hand the rule is applicable with match g to this P/T-system provided that a suitable gluing condition holds which is essential for the construction of the intermediate P/T-system.

Finally we evaluate the term *transform(r,m)* resulting in the direct transformation shown in Fig. 3, where we delete in a first step $g(L_2 \setminus I_2)$ from $(phi_3, table_{45})$ leading to P/T-system C. In a second step we glue together the P/T-systems C and R_2 along I_2 leading to P/T-system $table_{345}$ in Fig. 3, where the philosophers 3, 4, and 5 are sitting at the table, all of them in state *thinking*. The effect of firing the transition *enter restaurant* in the AHL-net in Fig. 1 with assignments of variables as discussed above is the removal of P/T-systems phi_3 from place *Entrance Hall* and $table_{45}$ from place *Restaurant* and adding the P/T-System $table_{345}$ to the place *Restaurant*.

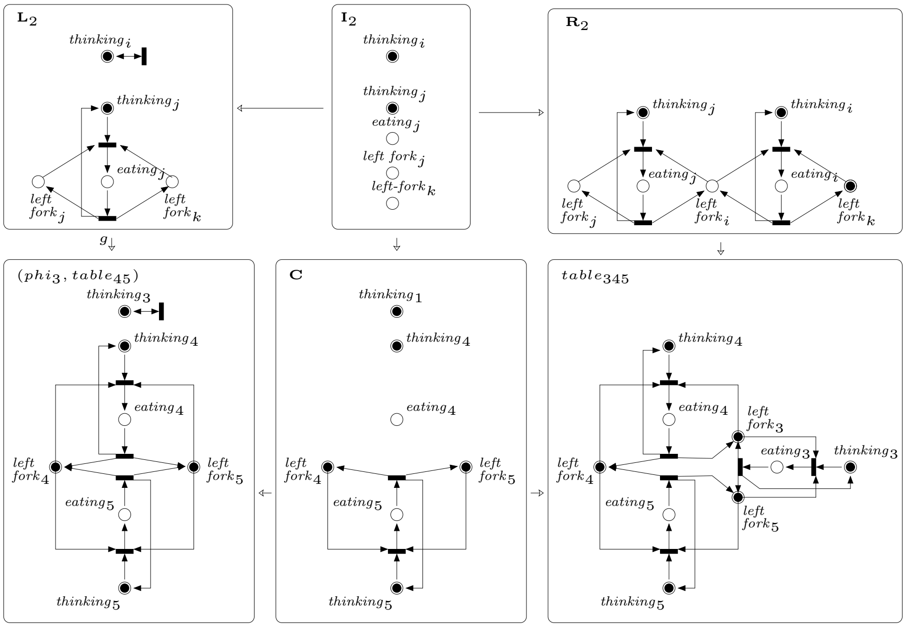

Fig. 3. Transformation of phi_3 and $table_{45}$ using rule $rule_2$

Consider now a different situation where the table $table'_{45}$ (see right hand side of Fig. 2) is in place *Restaurant* in Fig. 1. Using a different variable assignment v_3 the philosopher 3 sits down at table $table'_{45}$ where the philosopher 5 is in the state $eating_5$. In this case the variable n_1 is assigned to the table $table'_{45}$, the variable m to a suitable match morphism g' (i.e. from L_2 in Fig. 3 to $(phi_3, table'_{45})$ in Fig. 5

similar to g in Fig. 3) and the evaluation of the net inscription *transform(r,m)* of the transition *enter restaurant* in the AHL-net in Fig. 1 leads to the direct transformation depicted in Fig. 5. As a result we have now $table'_{345}$ on place *Restaurant*.

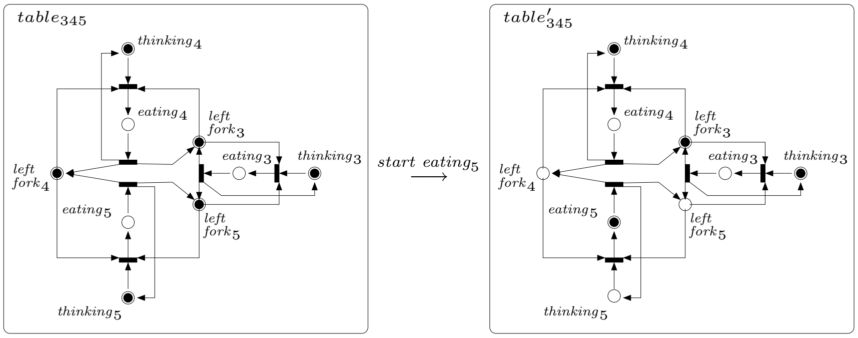

Fig. 4. P/T-systems $table_{345}$ and $table'_{345}$ of philosophers 3, 4 and 5

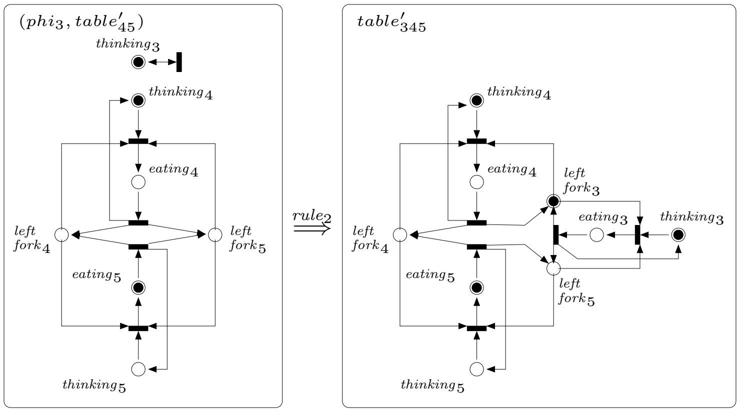

Fig. 5. Transformation of phi_3 and $table'_{45}$ using rule $rule_2$

Summarising, we have explained two different firing sequences of the AHL-net in Fig. 1. The first one starts with the token firing of $table_{45}$ leading to the P/T-system $table'_{45}$ (see Fig. 2) before philosopher 3 sits down at the table $table'_{45}$, so that we get the table $table'_{345}$ (see Fig. 5). The second one begins by philosopher 3 sitting down at table $table_{45}$ (see Fig. 3) before the philosopher 5 starts eating (see Fig. 4).

According to the spirit of processes for low-level nets we want to consider now processes for AHL-nets based on AHL-occurrence nets. In fact the two firing sequences considered above correspond to different AHL-occurrence nets. An AHL-occurrence net is similar to a low-level occurrence net concerning unitarity, conflict freeness, and acyclicity. However, in contrast to a low-level occurrence net an AHL-occurrence net realises more than one concurrent computation depending on different initial markings and variable assignments. For this reason we consider AHL-occurrence nets with a set of initial markings of the input places and corresponding instantiations of places and transitions by data and consistent variable assignments, respectively. For more details we refer to Section 3.

For the two different firing sequences we get the two different AHL-occurrence nets with initial markings $K_{Eat/Enter}$ and $K_{Enter/Eat}$ with corresponding instantiations $L_{Eat/Enter}$ and $L_{Enter/Eat}$ depicted in Fig. 6 and Fig. 7. Note that the AHL-occurrence nets $K_{Eat/Enter}$ and $K_{Enter/Eat}$ have the same input and output places as well as the same initial marking. But due to the firing of the transitions *start/stop activities* and *enter restaurant* in opposite order we use different variable evaluations v_1 and v_3 in $L_{Eat/Enter}$ and v_2 and v_4 in $L_{Enter/Eat}$, respectively. Nevertheless, the AHL-occurrence nets have the same input/output behaviour, i.e. the two different firing sequences end up with the same marking of the output places where the philosopher 3 sits together with the philosopher 4 and 5 at the table and philosopher 5 has started to eat (see table $table'_{345}$ on the right hand sides of Fig. 4 and Fig. 5).

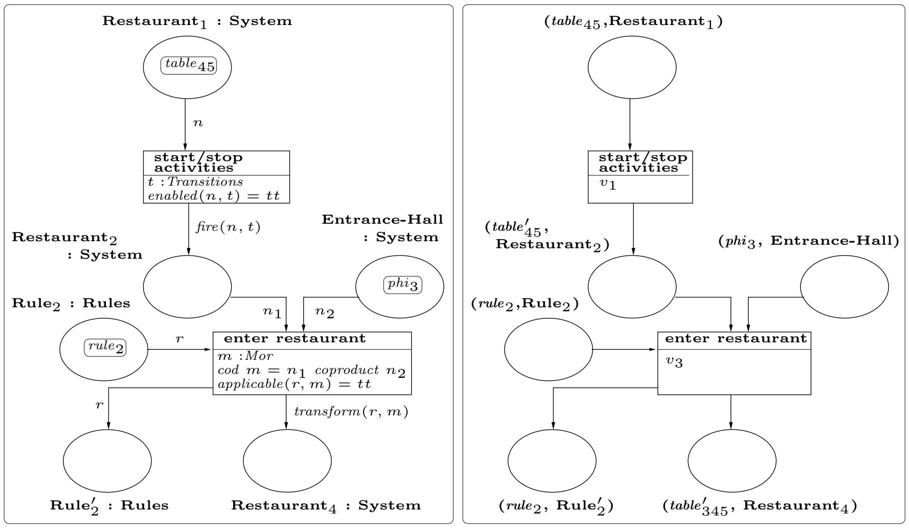

Fig. 6. AHL-occurrence net $K_{Eat/Enter}$ with instantiation $L_{Eat/Enter}$

Moreover we will show in Section 4 that there are basic AHL-occurrence nets K_{Eat} and K_{Enter}, s.t. $K_{Eat/Enter}$ and $K_{Enter/Eat}$ can be obtained as compo-

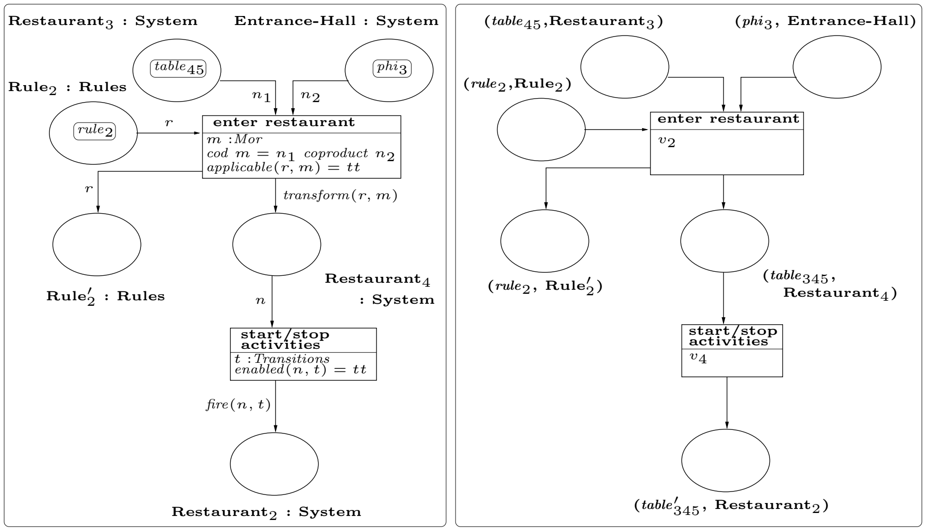

Fig. 7. AHL-occurrence net $K_{Enter/Eat}$ with instantiation $L_{Enter/Eat}$

sition in different order of K_{Eat} and K_{Enter}. This allows to consider the corresponding processes of K_{Eat} and K_{Enter} with instantiations as independent processes of the AHL-net AN_{House} in Fig. 1.

3 Algebraic High-Level Net Processes

In this section we review the concept of algebraic high-level nets and we give a formal definition of high-level processes [6, 7] based on high-level occurrence nets. Moreover we extend this definition by a suitable notation of instantiations for each initial marking. As net formalism we use place/transition nets following the notation of "Petri nets are Monoids" in [9].

Definition 1 (Place/Transition Net). *A place/transition (P/T) net $N = (P, T, pre, post)$ consists of sets P and T of places and transitions respectively, and pre- and post domain functions $pre, post : T \to P^{\oplus}$ where $P^{\oplus}$ is the free commutative monoid over P.*

A P/T-net morphism $f : N_1 \to N_2$ 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

$$f_P^{\oplus} \circ pre_1 = pre_2 \circ f_T \text{ and } f_P^{\oplus} \circ post_1 = post_2 \circ f_T$$

where the extension $f_P^{\oplus} : P_1^{\oplus} \to P_2^{\oplus}$ of $f_P : P_1 \to P_2$ is defined by $f_P^{\oplus}(\sum_{i=1}^{n} k_i \cdot p_i) = \sum_{i=1}^{n} k_i \cdot f_P(p_i)$. A P/T-net morphism $f = (f_P, f_T)$ is called injective if f_P and f_T are injective and is called isomorphism if f_P and f_T are bijective.

The category defined by P/T-nets and P/T-net morphisms is denoted by **PT-Net** *where the composition of P/T-net morphisms is defined componentwise for places and transitions.*

Because the notion of pushouts is essential for our main results we state the construction of pushouts in the category **PTNet** of place/transition nets. Intuitively a pushout means the gluing of two nets along an interface net. The construction is based on the pushouts for the sets of transitions and places in the category **SET**. In the category **SET** of sets and functions the pushout object D for given $f_1 : A \to B$ and $f_2 : A \to C$ is defined by the quotient set $D = B \uplus C / \equiv$, short $D = B \circ_A C$, where $B \uplus C$ is the disjoint union of B and C and $\equiv$ is the equivalence relation generated by $f_1(a) \equiv f_2(a)$ for all $a \in A$. In fact, D can be interpreted as the gluing of B and C along A: Starting with the disjoint union $B \uplus C$ we glue together the elements $f_1(a) \in B$ and $f_2(a) \in C$ for each $a \in A$.

The pushout object N_3 in the category **PTNet** is constructed componentwise for transitions and places in **SET** with corresponding pre- and post domain functions. For given P/T-net morphisms $f_1 : N_0 \to N_1$ and $f_2 : N_0 \to N_2$ the pushout of f_1 and f_2 is defined by the pushout diagram (PO) in **PTNet** and is denoted by $N_3 = N_1 \circ_{(N_0,f_1,f_2)} N_2$. For details we refer to [10].

Definition 2 (Pushouts of Place/Transition Nets). *Given P/T-net morphisms $f_1 : N_0 \to N_1$ and $f_2 : N_0 \to N_2$ then the pushout diagram (1) and the pushout object N_3 in the category* **PTNet**, *written $N_3 = N_1 \circ_{(N_0,f_1,f_2)} N_2$, with $N_x = (P_x, T_x, pre_x, post_x)$ for $x = 0, 1, 2, 3$ is constructed as follows:*

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

$$\begin{tikzcd}
N_0 \arrow[r, "f_1"] \arrow[d, "f_2"'] \arrow[dr, phantom, "(1)"] & N_1 \arrow[d, "f'_1"] & T_0 \arrow[r, "f_{1,T}"] \arrow[d, "f_{2,T}"'] \arrow[dr, phantom, "(2)"] & T_1 \arrow[d, "f'_{1,T}"] & P_0 \arrow[r, "f_{1,P}"] \arrow[d, "f_{2,P}"'] \arrow[dr, phantom, "(3)"] & P_1 \arrow[d, "f'_{1,P}"] \\
N_2 \arrow[r, "f'_2"'] & N_3 & T_2 \arrow[r, "f'_{2,T}"'] & T_3 & P_2 \arrow[r, "f'_{2,P}"'] & P_3
\end{tikzcd}$$

Two examples of the pushout construction of P/T-nets are depicted in Fig. 3, where the pushouts describes the gluing of the nets L_2 and C along I_2 and the gluing of the nets R_2 and C along I_2, respectively.

In the following we review the definition of AHL-nets from [6, 7].

Definition 3 (Algebraic High-Level Net). *An algebraic high-level (AHL) net $AN = (SP, P, T, pre, post, cond, type, A)$ consists of*

- *an algebraic specification $SP = (\Sigma, E; X)$ with signature $\Sigma = (S, OP)$, equations E, and 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, E)*-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 and AHL-net morphisms is denoted by **AHLNet** *where the composition of AHL-net morphisms is defined componentwise for places and transitions.*

In the following we omit the indices of functions f_P and f_T if no confusion arises. An example of an AHL-net is given in Section 2, where the "House of Philosophers" in Fig. 1 is an AHL-net with data type part consisting of the signature *HLRN-System-SIG* and algebra A according to [8] (see Appendix A.1).

The construction of pushouts in the category **AHLNet** of AHL-nets with fixed specification SP and algebra A can be analogously defined to the construction of pushouts in **PTNet** described above (for details see [10]).

Now we introduce high-level occurrence nets and processes according to [6,7]. 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 computation due to different initial markings. In fact, high-level occurrence nets can be considered to have a set of initial markings for the input places, whereas there is only one implicit initial marking of the input places for low-level occurrence nets.

Definition 4 (AHL-Occurrence Net). *An AHL-occurrence net* K *is an AHL -net* $K = (SP, P, T, pre, post, cond, type, A)$ *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* $< \subseteq (P \times T) \cup (T \times P)$ *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 finitary strict partial order, i.e. the partial order is irreflexive and for each element in the partial order the set of its predecessors is finite.*

The notion of high-level net processes generalises the one of low-level net processes, where a P/T-process of a P/T-net N is a P/T-net morphism $p : K \rightarrow N$ and K is a low-level occurrence net, i.e. a net satisfying conditions 1.-4. in Def. 4. Examples of high-level and low-level occurrence nets are given in Fig. 6 and Fig. 7 in Section 2.

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

Because in general there are different meaningful markings of an AHL-occurrence net K, we introduce a set of initial markings of the input places of K.

Definition 6 (AHL-Occurrence Net with Initial Markings). *An AHL-occurrence net with initial markings $(K, INIT)$ consists of an AHL-occurrence net K and a set $INIT$ of initial markings $init \in INIT$ of the input places $IN(K)$, where the input places of K are defined by $IN(K) = \{p \in P | \bullet p = \emptyset\}$ and similarly the output places of K are defined by $OUT(K) = \{p \in P | p\bullet = \emptyset\}$.*

The following notion of instantiation defines one concurrent execution of a marked high-level occurrence net. In more detail an instantiation is a subnet of the flattening of the AHL-occurrence net corresponding to the initial marking. In [6,7] it is shown that for a marked AHL-occurrence net there exists a complete firing sequence if and only if there exists an instantiation which net structure is isomorphic to the AHL-occurrence net and has the initial marking of the AHL-occurrence net as input places. Note that in general we may have different instantiations for the same initial marking.

The flattening $Flat(AN)$ of an AHL-net AN results in a corresponding low-level net N, where the data type part (SP, A) and the firing behaviour of the AHL-net AN is encoded in the sets of places and transitions of N. Thus the flattening $Flat(AN)$ leads to an infinite P/T-net N if the algebra A is infinite. In contrast the skeleton $Skel(AN)$ of an AHL-net AN is a low-level net N' preserving the net structure of the AHL-net but dropping the net inscriptions. While there is a bijective correspondence between firing sequences of the AHL-net and firing sequences of its flattening, each firing of the AHL-net implies a firing of the skeleton, but not vice versa. For details we refer to [6, 7] and to the Appendix A.2.

Definition 7 (Instantiations of AHL-Occurrence Net). *Given an AHL-occurrence net with initial markings $(K, INIT)$ with $init \in INIT$. An instantiation L_{init} of $(K, init)$ is a low-level occurrence net $L_{init} \subseteq Flat(K)$ with input places $IN(L_{init}) = init$ such that the projection $proj : L_{init} \rightarrow Skel(K)$ defined by $proj_P(a, p) = p$ and $proj_T(t, v) = t$ is an isomorphism of low-level occurrence nets.*

As mentioned above for a given initial marking of an AHL-occurrence net there exist in general more than one instantiation and thus different firing sequences resulting in different markings of the output places of the AHL-occurrence net. For this reason we introduce the new notion of AHL-occurrence nets and AHL-processes with instantiations, where we fix exactly one instantiation for a given initial marking, i.e. one concurrent execution of the marked AHL-occurrence net.

Definition 8 (AHL-Occurrence Net with Instantiations). *An AHL-occurrence net with instantiations $KI = (K, INIT, INS)$ is an AHL-occurrence net with initial markings $(K, INIT)$ and a set INS of instantiations, such that for each init $\in INIT$ we have a distinguished instantiation $L_{init} \in INS$, i.e. $INS = \{L_{init} | init \in INIT\}$.*

An AHL-occurrence net with instantiations KI defines for each init $\in INIT$ with $IN(L_{init}) = init$ an output $out = OUT(L_{init})$ with $proj_P(out) = OUT(K)$. Let $EXIT$ be the set of all markings of the output places $OUT(K)$, then we obtain a function $inout : INIT \rightarrow EXIT$ by $inout(init) = OUT(L_{init})$.

Definition 9 (AHL-Process with Instantiations). *An instantiated AHL-process of an AHL-net AN is an AHL-occurrence net with instantiations $KI = (K, INIT, INS)$ together with an AHL-net morphism $mp : K \rightarrow AN$.*

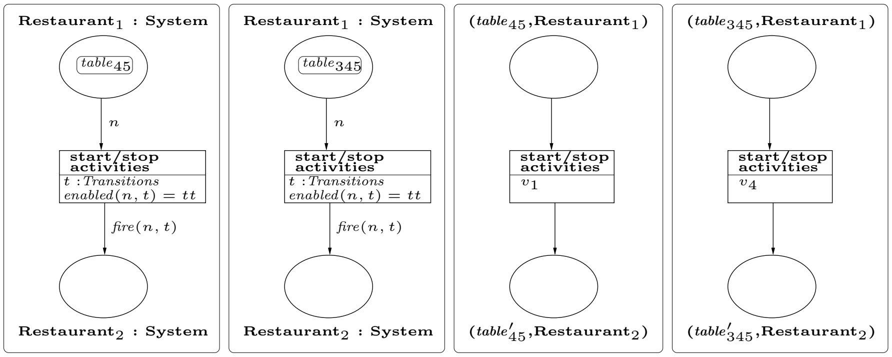

Fig. 8. AHL-occurrence net K_{Eat} with instantiations L_{Eat} and $L_{Eat'}$

As an example the AHL-occurrence net with instantiations $KI_{Eat} = (K_{Eat}, INIT_{Eat}, INS_{Eat})$ is depicted in Fig. 8 according to the discussion in Section 2. The AHL-occurrence net K_{Eat} is the AHL-net on the left hand side of Fig. 8 without the marking of the place *Restaurant*$_1$. There are two different initial markings, i.e the set of initial markings is defined by $INIT_{EAT} = \{(table_{45}, Restaurant_1), (table_{345}, Restaurant_1)\}$ and the set of instantiations by $INS_{Eat} = \{L_{Eat}, L_{Eat'}\}$ (see the two instantiations on the right hand side of Fig. 8).

The instantiated AHL-process is the AHL-occurrence net with instantiations KI_{Eat} together with the AHL-net morphism $mp_{Eat} : K_{Eat} \rightarrow AN_{House}$. The morphism mp_{Eat} consists of an obvious inclusion of the transition *start/stop activities*, while the places named $Restaurant_1$ and $Restaurant_2$ are mapped to the place *Restaurant* of the AHL-net AN_{House} in Fig. 1.

Further examples are given in Section 2, where in Fig. 6 we have the AHL-occurrence net with instantiations $KI_{Eat/Enter}$ and in Fig. 7 the AHL-occurrence net with instantiations $KI_{Enter/Eat}$.

4 Composition, Equivalence and Independence of Algebraic High-Level Net Processes

Based on the construction of pushouts of low-level and high-level nets introduced in the previous section we define in this section the composition of AHL-occurrence nets and AHL-processes with instantiations and we introduce the concept of equivalence and independence of high-level net processes. Two independent high-level net processes can be composed in any order leading to equivalent high-level net processes which especially have the same input/output behaviour.

The composition of two AHL-occurrence nets K_1 and K_2 is defined by merging some of the output places of K_1 with some of the input places of K_2, so that the result of the composition definitely is an AHL-occurrence net. In general the composition of AHL-occurrence nets is not an AHL-occurrence net, because the result of gluing two high-level occurrence nets may contain forward and/or backward conflicts as well as the partial order might be violated. Thus we state suitable conditions, so that the composition of AHL-occurrence nets leads to an AHL-occurrence net. Moreover we generalise this construction on the one hand to corresponding instantiations and on the other hand to AHL-net morphisms, so that the composition of AHL-processes with instantiation leads to an AHL-process under suitable conditions.

Definition 10 (Composability of AHL-Occurrence Nets). *Given the AHL-occurrence nets* $K_x = (SP, P_x, T_x, pre_x, post_x, cond_x, type_x, A)$ *for* $x = 1, 2$ *and* $I = (SP, P_I, T_I, pre_I, post_I, cond_I, type_I, A)$ *with* $T_I = \emptyset$ *and two injective AHL-net morphisms* $i_1 : I \to K_1$ *and* $i_2 : I \to K_2$*. Then* (K_1, K_2) *is composable w.r.t.* (I, i_1, i_2) *if* $i_1(P_I) \subseteq OUT(K_1)$ *and* $i_2(P_I) \subseteq IN(K_2)$*.*

Theorem 1 (Composition of AHL-Occurrence Nets). *Given the AHL-occurrence nets* K_1, K_2 *and* I *as above and two injective AHL-net morphisms* $i_1 : I \to K_1$ *and* $i_2 : I \to K_2$ *such that* (K_1, K_2) *is composable w.r.t.* (I, i_1, i_2)*. Then the pushout digram (PO) exists in the category* **AHLNet** *and the pushout object* K*, with* $K = K_1 \circ_{(I,i_1,i_2)} K_2$*, is an AHL-occurrence net and is called composition of* (K_1, K_2) *w.r.t.* (I, i_1, i_2)*.*

$$\begin{tikzcd}
I \arrow[r, "i_1"] \arrow[d, "i_2"'] \arrow[dr, phantom, "\textbf{(PO)}"] & K_1 \arrow[d, "i_1'"] \\
K_2 \arrow[r, "i_2'"'] & K
\end{tikzcd}$$

Proof. (Sketch) The detailed proof can be found in Appendix A.3.

For the existence and construction of pushouts in **AHLNet** we refer to [10]. As mentioned in Section 3 it can be constructed componentwise similar to pushouts in **PTNet**. It remains to show that the result of the composition of (K_1, K_2) w.r.t. (I, i_1, i_2) given by $K = (SP, P, T, pre, post, cond, type, A)$ is an occurrence net indeed:

1. *Unarity:* is obtained as the set of transitions T is obtained by disjoint union.
2. *No forward conflicts:* Since AHL-net morphisms preserve the adjacencies of transitions (i.e. pre and post domain), in case of $t_1 \neq t_2$ and $p \in \bullet t_1 \cap \bullet t_2$ for $t_1, t_2 \in T$ both transitions have a preimage in T_1 and T_2, respectively. Moreover, p has a preimage in both P_1 and P_2, so one of the preimages is in the corresponding OUT set. But this contradicts the fact that this place has to be in the preset of the corresponding transition.
3. *No backward conflicts:* Analogously.
4. *Partial Order:* follows from the partial order of K_1 and K_2 and from the composability condition.

Note that the order of K_1 and K_2 in the pair (K_1, K_2) and the result $K = K_1 \circ_{(I,i_1,i_2)} K_2$ is important because i_1 and i_2 relate output places of K_1 with input places K_2. The composition of two AHL-occurrence nets is called strict parallel if $P_I = \emptyset$ and is called strict sequential if $i_1(P_I) = OUT(K_1)$ and $i_2(P_I) = IN(K_2)$.

Definition 11 (Composition of Instantiations). *Given the AHL-occurrence nets K_1, K_2 and I as above and two injective AHL-net morphism $i_1 : I \to K_1$ and $i_2 : I \to K_2$ such that (K_1, K_2) is composable w.r.t. (I, i_1, i_2). Let $KI_x = (K_x, INIT_x, INS_x)$ for $x = 1, 2$ be two AHL-occurrence nets with instantiations and $L_{init_1} \in INS_1$ and $L_{init_2} \in INS_2$. Then (L_{init_1}, L_{init_2}) is composable w.r.t. (I, i_1, i_2) if for all $(a, p) \in A_{type(p)} \otimes P_I : (a, i_1(p)) \in OUT(L_{init_1}) \Rightarrow (a, i_2(p)) \in IN(L_{init_2})$.*

From (I, i_1, i_2) we construct the induced instantiation interface (J, j_1, j_2) of (L_{init_1}, L_{init_2}) with $J = (P_J, T_J, pre_J, post_J)$ by

- *$P_J = \{(a, p) | (a, i_1(p)) \in OUT(L_{init_1})\}$,*
- *$T_J = \emptyset$,*
- *$pre_J = post_J = \emptyset$ (the empty function) and*
- *$j_x : J \to L_{init_x}$ for $x = 1, 2$ defined by $j_{x,P} = id_A \otimes i_{x,P}$ and $j_{x,T} = \emptyset$.*

$$\begin{tikzcd} J \arrow[r, "j_1"] \arrow[d, "j_2"'] \arrow[dr, phantom, "\mathbf{(PO)}"] & L_{init_1} \arrow[d, "j_1'"] \\ L_{init_2} \arrow[r, "j_2'"'] & L_{init} \end{tikzcd}$$

The composition of (L_{init_1}, L_{init_2}) w.r.t. the instantiation interface (J, j_1, j_2) induced by (I, i_1, i_2) is defined by the pushout diagram (PO) in **PTNet** *and is denoted by $L_{init} = L_{init_1} \circ_{(J,j_1,j_2)} L_{init_2}$.*

The AHL-occurrence net with instantiations $KI_{Enter} = (K_{Enter}, INIT_{Enter}, INS_{Enter})$ is given in Figs. 9 and 10. The sequential composition of K_{Eat} (see Fig. 8 in Section 3) and K_{Enter} is defined by merging the output place $Restaurant_2$ of K_{Eat} and the input place $Restaurant_3$ of K_{Enter} leading to the AHL-occurrence net $K_{Eat/Enter}$ (see Fig. 6 in Section 2). In more detail $K_{Eat/Enter} = K_{Eat} \circ_{(I,i_1,i_2)} K_{Enter}$ is the gluing of the two basic AHL-occurrence nets along I with $P_I = \{Restaurant\}$, $i_1(Restaurant) = Restaurant_2$

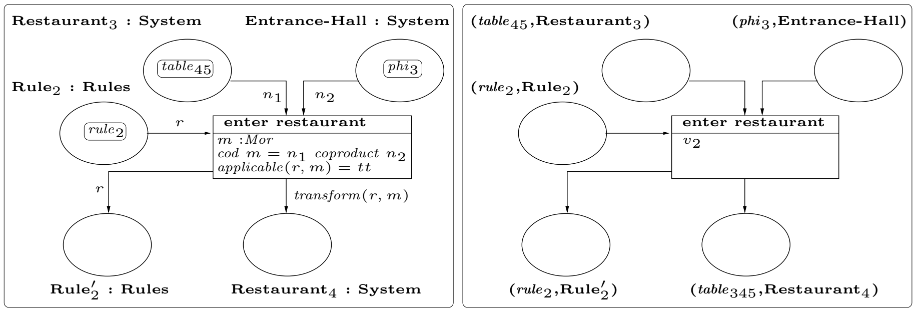
$\mathbf{Restaurant_3}$: System
Entrance-Hall : System
$table_{45}$
phi_3
$\mathbf{Rule_2}$: Rules
n_1
n_2
$rule_2$
r
enter restaurant
m :Mor
$cod\ m = n_1\ coproduct\ n_2$
$applicable(r, m) = tt$
r
$transform(r, m)$
$\mathbf{Rule'_2}$: Rules
$\mathbf{Restaurant_4}$: System
($table_{45}$,$\mathbf{Restaurant_3}$)
(phi_3,Entrance-Hall)
($rule_2$,$\mathbf{Rule_2}$)
enter restaurant
v_2
($rule_2$,$\mathbf{Rule'_2}$)
($table_{345}$,$\mathbf{Restaurant_4}$)

Fig. 9. AHL-occurrence net K_{Enter} with instantiation L_{Enter}

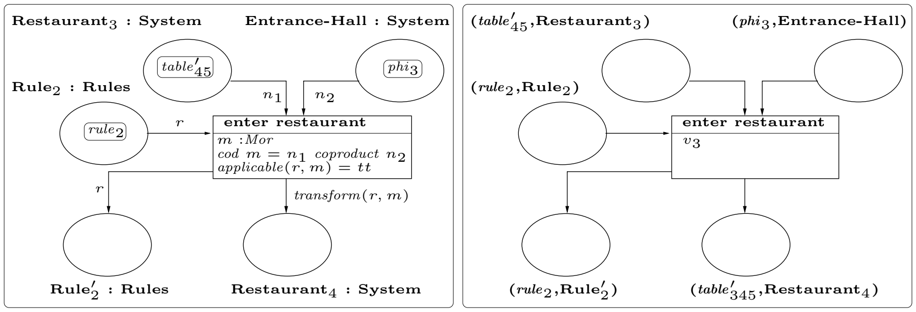
$\mathbf{Restaurant_3}$: System
Entrance-Hall : System
$table'_{45}$
phi_3
$\mathbf{Rule_2}$: Rules
n_1
n_2
$rule_2$
r
enter restaurant
m :Mor
$cod\ m = n_1\ coproduct\ n_2$
$applicable(r, m) = tt$
r
$transform(r, m)$
$\mathbf{Rule'_2}$: Rules
$\mathbf{Restaurant_4}$: System
($table'_{45}$,$\mathbf{Restaurant_3}$)
(phi_3,Entrance-Hall)
($rule_2$,$\mathbf{Rule_2}$)
enter restaurant
v_3
($rule_2$,$\mathbf{Rule'_2}$)
($table'_{345}$,$\mathbf{Restaurant_4}$)

Fig. 10. AHL-occurrence net K_{Enter} with instantiation $L_{Enter'}$

and $i_2(\mathit{Restaurant}) = \mathit{Restaurant}_3$. The corresponding instantiations L_{Eat} in Fig. 8 and $L_{Enter'}$ in Fig. 10 can be analogously composed to the instantiation $L_{Eat/Enter}$ in Fig. 6. Note that $(L_{Eat}, L_{Enter'})$ is composable, because we have $(\mathit{table}'_{45}, i_1(\mathit{Restaurant})) \in OUT(L_{Eat})$ and $(\mathit{table}'_{45}, i_2(\mathit{Restaurant})) \in IN(L_{Enter'})$.

Theorem 2 (Composition of AHL-Occurrence Nets with Instantiations). *Given the AHL-occurrence nets K_1, K_2 and I as above and two injective AHL-net morphism $i_1 : I \to K_1$ and $i_2 : I \to K_2$ such that (K_1, K_2) is composable w.r.t. (I, i_1, i_2). Let $KI_x = (K_x, INIT_x, INS_x)$ for $x = 1, 2$ be two AHL-occurrence nets with instantiations. Then the composition of (KI_1, KI_2) w.r.t. (I, i_1, i_2) is defined by $KI = (K, INIT, INS)$ with*

- *$K = K_1 \circ_{(I,i_1,i_2)} K_2$,*
- *$INS = \{L_{init_1} \circ_{(J,j_1,j_2)} L_{init_2} | L_{init_x} \in INS_x$ for $x = 1, 2, (L_{init_1}, L_{init_2})$ is composable w.r.t. (J, j_1, j_2) induced by $(I, i_1, i_2)\}$,*
- *and $INIT = \{IN(L_{init}) | L_{init} \in INS\}$*

and $KI = KI_1 \circ_{(I,i_1,i_2)} KI_2$ is an AHL-occurrence net with instantiations.

Proof. (Sketch) The detailed proof can be found in Appendix A.4.

To prove that $KI = (K, INIT, INS)$ is well-defined, first note that K is an occurrence net due to Theorem 1. Moreover, for each $L_{init} \in INS$ we need to show $L_{init} \subseteq \mathit{Flat}(K)$ and $in \circ \mathit{proj}(K) : L_{init} \to \mathit{Skel}(K)$ is an isomorphism in the diagram below where we have the following pushouts: (PO1) by construction, (PO2) since Flat preserves pushout $K = K_1 \circ_I K_2$ and (PO3) since Skel preserves pushout $K = K_1 \circ_I K_2$.

$\mathit{proj}(I), \mathit{proj}(K_1), \mathit{proj}(K_2)$ and $\mathit{proj}(K)$ are projections from the flattening to the skeleton construction (see Remark 1 in Appendix A.2) and in_I, in_1, in_2 are inclusions where $J \subseteq \mathit{Flat}(I) = (A \otimes P, \emptyset, \emptyset, \emptyset)$ and in is induced by (PO1).

$$\begin{tikzcd}
 & L_{init1} \arrow[rr, "in_1"] \arrow[ddr] \arrow[r, phantom, "\mathbf{(PB)}"] & & \mathit{Flat}(K_1) \arrow[rr, "\mathit{proj}(K_1)"] \arrow[ddr] & & \mathit{Skel}(K_1) \arrow[ddr] & \\
J \arrow[ur, "j_1"] \arrow[rr, "in_I"] \arrow[ddr, "j_2"'] \arrow[dr, phantom, "\mathbf{(PO1)}"] & & \mathit{Flat}(I) \arrow[ur, "\mathit{Flat}(i_1)"] \arrow[rr, "\mathit{proj}(K_1)"] \arrow[ddr, "\mathit{Flat}(i_2)"] \arrow[dr, phantom, "\mathbf{(PO2)}"] & & \mathit{Skel}(I) \arrow[ur] \arrow[ddr] \arrow[dr, phantom, "\mathbf{(PO3)}"] & & \\
 & & L_{init} \arrow[rr, "in"] & & \mathit{Flat}(K) \arrow[rr, "\mathit{proj}(K)"] & & \mathit{Skel}(K) \\
 & L_{init2} \arrow[ur] \arrow[rr, "in_2"] & & \mathit{Flat}(K_2) \arrow[ur] \arrow[rr, "\mathit{proj}(K_2)"] & & \mathit{Skel}(K_2) \arrow[ur] &
\end{tikzcd}$$

Since $\mathit{proj}(I) \circ in_I$ can be shown to be an isomorphism (using that J is pullback of $\mathit{Flat}(i_1)$ and in_1) and $\mathit{proj}(K_x) \circ in_x$ are by assumption isomorphisms for $x = 1, 2$, we conclude that $\mathit{proj}(K) \circ in$ is isomorphic as well. Hence in is injective and can be chosen to be an inclusion $in : L_{init} \to \mathit{Flat}(K)$.

Given the two basic AHL-occurrence nets with instantiations KI_{Eat} and KI_{Enter}, then the composition of (KI_{Eat}, KI_{Enter}) results in the AHL-occurrence net with instantiation $KI_{Eat/Enter}$ (see Fig. 6 in Section 2), while the opposite composition of (KI_{Enter}, KI_{Eat}) is the AHL-occurrence net with instantiation $KI_{Enter/Eat}$ (see Fig. 7 in Section 2). Different to INS_{Eat} and INS_{Enter} the set of instantiations $INS_{Eat/Enter}$ only consists of one instantiation $L_{Eat/Enter}$ (and analogously $INS_{Enter/Eat}$), because we require in Theorem 2 the composability of instantiations.

Definition 12 (Composability of AHL-Processes with Instantiations). *Given the AHL-occurrence nets K_1, K_2 and I as above and two injective AHL-net morphism $i_1 : I \to K_1$ and $i_2 : I \to K_2$. Let $KI_x = (K_x, INIT_x, INS_x)$ together with the AHL-net morphisms $mp_x : K_x \to AN$ for $x = 1, 2$ be two instantiated AHL-processes of the AHL-net AN. Then (mp_1, mp_2) is composable w.r.t. (I, i_1, i_2) if*

1. *(K_1, K_2) is composable w.r.t. (I, i_1, i_2) and*
2. *$mp_1 \circ i_1 = mp_2 \circ i_2$.*

Theorem 3 (Composition of AHL-Processes with Instantiations). *Given the AHL-occurrence nets K_1, K_2 and I as above and two injective AHL-net morphism $i_1 : I \to K_1$ and $i_2 : I \to K_2$. Let $KI_x = (K_x, INIT_x, INS_x)$ together with the AHL-net morphisms $mp_x : K_x \to AN$ for $x = 1, 2$ be two instantiated AHL-processes of the AHL-net AN such that (mp_1, mp_2) is composable w.r.t. (I, i_1, i_2). Then the instantiated AHL-occurrence net $KI = KI_1 \circ_{(I,i_1,i_2)} KI_2$ together with the induced AHL-net morphism $mp : K \to AN$ is an instantiated AHL-process of the AHL-net AN, where K is the AHL-occurrence net of KI.*

$$\begin{tikzcd}
I \arrow[r, "i_1"] \arrow[d, "i_2"'] \arrow[dr, phantom, "(\mathbf{PO})"] & K_1 \arrow[d, "i'_1"] \arrow[ddr, bend left, "mp_1"] & \\
K_2 \arrow[r, "i'_2"'] \arrow[drr, bend right, "mp_2"'] & K \arrow[dr, "m_p"] & \\
 & & AN
\end{tikzcd}$$

Proof. (Sketch) The detailed proof can be found in Appendix A.5.

Due to Def. 12 and the universal property of pushouts there is the morphism $m_P : K \to AN$, that uniquely commutes $mp_1 = i'_1 \circ m_P$ and $mp_2 = i'_2 \circ m_P$.

Because for low-level occurrence nets the input/output behaviour is fixed by the net structure, two low-level occurrence nets should be considered to be equivalent if they are isormorphic. For high-level occurrence nets the input/output behaviour additionally depends on the marking of their input places and on corresponding variable assignments. Hence we introduce the equivalence of two AHL-processes with instantiations, where the net structures of equivalent AHL-processes may be different, but they have especially the same input/output behaviour.

In more detail they have (up to renaming) the same sets of transitions and places and their instantiations are equivalent, i.e. there exist corresponding instantiations with the same input/output behaviour. So specific firing sequences of equivalent AHL-processes are comparable in the sense that they start and end up with the same data elements as marking of their input places and output places, respectively, but in general the corresponding transitions are fired in a different order.

Definition 13 (Equivalence of AHL-Processes with Instantiations). *Let $KI = (K, INIT, INS)$ and $KI' = (K', INIT', INS')$ together with AHL-net* *morphisms $mp : K \to AN$ and $mp' : K' \to AN$ two AHL-processes of an AHL-net AN. Then these two processes are called equivalent if*

1. *there are bijections $e_P : P_K \to P_{K'}$ and $e_T : T_K \to T_{K'}$ such the following diagram commutes componentwise*

$$\begin{tikzcd}
K \arrow[rr, shift left, "e_P"] \arrow[rr, shift right, "e_T"'] \arrow[dr, bend right, "mp"'] & \arrow[d, phantom, "="] & K' \arrow[dl, bend left, "mp'"] \\
& AN &
\end{tikzcd}$$

2. *and the instantiations are equivalent, i.e. for each $L_{init} \in INS$ there exists a $L_{init'} \in INS'$ and vice versa such that*

$$\forall (a,p) \in A_{type(p)} \otimes P_K : (a,p) \in IN(L_{init}) \Leftrightarrow (a, e_P(p)) \in IN(L_{init'}) \text{ and } \\ (a,p) \in OUT(L_{init}) \Leftrightarrow (a, e_P(p)) \in OUT(L_{init'})$$

The equivalence of the instantiations means that there is a bijection between the input places $IN(K)$ and $IN(K')$ (resp. output places $OUT(K)$ and $OUT(K\prime)$), s.t. the input-output function $inout : INIT \to EXIT$ of KI and $inout' : INIT' \to EXIT'$ of KI' are equal up to bijection of input and output places. But it is not required that $e = (e_P, e_T) : K \to K'$ is an isomorphism, i.e. in general $e = (e_P, e_T)$ is not compatible with pre- and post domains.

In Section 2 the AHL-processes with instantiations $KI_{Eat/Enter}$ in Fig. 6 and $KI_{Enter/Eat}$ in Fig. 7 together with the obvious AHL-net morphisms $mp_1 : KI_{Eat/Enter} \to AN_{House}$ and $mp_2 : KI_{Eat/Enter} \to AN_{House}$ are equivalent. There is a bijection between their transitions and places, which is not an isomorphism. The bijection of places is defined by mapping the input places of $KI_{Eat/Enter}$ to the input places of $KI_{Enter/Eat}$ (and analogously the output places) and the place *Restaurant*$_2$ of $KI_{Eat/Enter}$ to the place *Restaurant*$_4$ of $KI_{Enter/Eat}$, such that the diagram in Def. 13 commutes componentwise. Moreover the instantiations $L_{Eat/Enter}$ in Fig. 6 and $L_{Enter/Eat}$ are equivalent.

The main result in this context are suitable conditions s.t. AHL-net processes with instantiation can be composed in any order leading to equivalent high-level net processes. Here we use especially the assumption that the instantiations are consistent, i.e. there is a close relation between their input and output places.

Definition 14 (Consistency of Instantiations). *Given AHL-occurrence nets K_1, K_2 and I as in Def. 10 and injective AHL-net morphism $i_1 : I \to K_1$, $i_2 : I \to K_2$, $i_3 : I \to K_1$ and $i_4 : I \to K_2$ such that (K_1, K_2) is composable w.r.t. (I, i_1, i_2) and (K_2, K_1) is composable w.r.t. (I, i_4, i_3) with pushout (1) and (2), respectively. Moreover let $KI_x = (K_x, INIT_x, INS_x)$ be AHL-occurrence nets with instantiations for K_x $(x = 1, 2)$.*

Then (INS_1, INS_2) is called consistent if for all composable $(L_{init_1}, L_{init_2}) \in INS_1 \times INS_2$ w.r.t. (J, j_1, j_2) induced by (I, i_1, i_2) with pushout (3) there are composable $(L_{init'_2}, L_{init'_1}) \in INS_2 \times INS_1$ w.r.t. (J, j_4, j_3) induced by (I, i_4, i_3) with pushout (4) and vice versa, s.t. in both cases the instantiations satisfy the following properties 1.-4. for gluing points GP defined below:

1. *$IN(L_{init_x}) \setminus GP(L_{init_x}) = IN(L_{init'_x}) \setminus GP(L_{init'_x})$ and*
2. *$OUT(L_{init_x}) \setminus GP(L_{init_x}) = OUT(L_{init'_x}) \setminus GP(L_{init'_x})$ for $x = 1, 2$*

Moreover we require for all $(a, p) \in A_{type(p)} \otimes P_I$:

3. $(a, i_3(p)) \in IN(L_{init_1}) \Leftrightarrow (a, i_2(p)) \in IN(L_{init'_2})$
4. $(a, i_1(p)) \in OUT(L_{init'_1}) \Leftrightarrow (a, i_4(p)) \in OUT(L_{init_2})$

The gluing points GP are defined by

- *$GP(P_{K_1}) = i_1(P_I) \cup i_3(P_I)$, $GP(P_{K_2}) = i_2(P_I) \cup i_4(P_I)$,*
- *$GP(L_{init_x}) = \{(a, p) \in L_{init_x} | p \in GP(P_{K_x})\}$ and*
- *$GP(L_{init'_x}) = \{(a, p) \in L_{init'_x} | p \in GP(P_{K_x})\}$ for $x = 1, 2$.*

$$\begin{tikzcd}
I \arrow[r, "i_1"] \arrow[d, "i_2"'] \arrow[dr, phantom, "(\mathbf{1})"] & K_1 \arrow[d, "i'_1"] \\
K_2 \arrow[r, "i'_2"'] & K
\end{tikzcd}
\quad
\begin{tikzcd}
I \arrow[r, "i_4"] \arrow[d, "i_3"'] \arrow[dr, phantom, "(\mathbf{2})"] & K_2 \arrow[d, "i'_4"] \\
K_1 \arrow[r, "i'_3"'] & K
\end{tikzcd}
\quad
\begin{tikzcd}
J \arrow[r, "j_1"] \arrow[d, "j_2"'] \arrow[dr, phantom, "(\mathbf{3})"] & L_{init_1} \arrow[d, "j'_1"] \\
L_{init_2} \arrow[r, "j'_2"'] & L_{init}
\end{tikzcd}
\quad
\begin{tikzcd}
J \arrow[r, "j_4"] \arrow[d, "j_3"'] \arrow[dr, phantom, "(\mathbf{4})"] & L_{init'_2} \arrow[d, "j'_4"] \\
L_{init'_1} \arrow[r, "j'_3"'] & L_{init'}
\end{tikzcd}$$

Let KI_{Eat} and KI_{Enter} be the two instantiated AHL-processes as described above. Their sets of instantiations INS_{Eat} and INS_{Enter} are consistent, because for the composable instantiations $(L_{Eat}, L_{Enter'}) \in INS_{Eat} \times INS_{Enter}$ there are the composable instantiations $(L_{Enter}, L_{Eat'}) \in INS_{Enter} \times INS_{Eat}$ (and vice versa) satisfying the properties 1.-4. in Def. 14.

Theorem 4 (Equivalence and Independence of AHL-Processes). *Given an AHL-net AN and AHL-occurrence nets $KI_x = (K_x, INIT_x, INS_x)$ with consistent instantiations as in Def. 14 with AHL-net morphisms $mp_x : K_x \to AN$ for $x = 1, 2$.*

Then we have instantiated AHL-processes $KI = (K, INIT, INS)$ with $mp : K \to AN$ and $KI' = (K', INIT', INS')$ with $mp' : K' \to AN$ defined by opposite compositions $KI = KI_1 \circ_{(I, i_1, i_2)} KI_2$ and $KI' = KI_2 \circ_{(I, i_4, i_3)} KI_1$ and both are equivalent processes of AN, provided that

1. *K_1 and K_2 have no isolated places, i.e. $IN(K_x) \cap OUT(K_x) = \emptyset$ for $x = 1, 2$*
2. *mp_1 and mp_2 are compatible with i_1, i_2, i_3 and i_4, i.e. $mp_1 \circ i_1 = mp_2 \circ i_2 = mp_1 \circ i_3 = mp_2 \circ i_4 : I \to AN$.*

Under these conditions KI_1 and KI_2 are called independent.

Proof. (Sketch) The detailed proof can be found in Appendix A.6. The instantiated AHL-processes KI and KI' with $mp : K \to AN$ and $mp' : K' \to AN$ exist by Theorem 3. It remains to show that they are equivalent.
Construction of bijections. The bijection $e_T : T_K \to T_{K'}$ follows from the fact that $I_T = \emptyset$ and hence $T_K \cong T_{K_1} \uplus T_{K_2}$ and $T_{K'} \cong T_{K_2} \uplus T_{K_1}$. In order to obtain the bijection $e_P : P_K \to P_{K'}$ we show that P_K and $P_{K'}$ can be represented by

the following disjoint unions of gluing points GP and non gluing points NGP in pushout (1) and (2) in Def. 14.

$$P_K = GP_1(P_K) \cup GP_2(P_K) \cup GP_3(P_K) \cup NGP(P_K) \text{ with}$$
$$GP_1(P_K) = i'_1 \circ i_3(P_I), GP_2(P_K) = i'_2 \circ i_4(P_I) \text{ and } GP_3(P_K) = i'_1 \circ i_1(P_I)$$

$$P_{K'} = GP_1(P_{K'}) \cup GP_2(P_{K'}) \cup GP_3(P_{K'}) \cup NGP(P_{K'}) \text{ with}$$
$$GP_1(P_{K'}) = i'_4 \circ i_2(P_I), GP_2(P_{K'}) = i'_3 \circ i_1(P_I) \text{ and } GP_3(P_{K'}) = i'_3 \circ i_3(P_I)$$

This allows to define $e_{P_x} : GP_x(P_K) \to GP_x(P_{K'})$ for $x = 1, 2, 3$ by $e_{P_1}(i'_1 \circ i_3(p)) = i'_4 \circ i_2(p)$ for all $p \in P_I$ and similar for e_{P_2} and e_{P_3}. Since i'_1, i_3, i'_4, and i_2 are all injective e_{P_1} is bijective and similar also e_{P_2} and e_{P_3} are bijective.

Finally also $e_{P_4} : NGP(P_K) \to NGP(P_{K'})$ can be defined as bijection. Using $IN(K_x) \cap OUT(K_x) = \emptyset$ for $x = 1, 2$ it can be shown that P_K (and similar $P_{K'}$) is a disjoint union of all four components leading to a bijection $e_P = e_{P_1} \cup e_{P_2} \cup e_{P_3} \cup e_{P_4} : P_K \to P_{K'}$. With these definitions it can be shown explicitly that the diagram in Def. 13 commutes componentwise.

Equivalence of instantiations. Given $L_{init} = L_{init_1} \circ_{(J,j_1,j_2)} L_{init_2}$ with pushout (3) in Def. 14 we have by consistency of (INS_1, INS_2) $L_{init'} = L_{init'_2} \circ_{(J,j_4,j_3)} L_{init'_1}$ with pushout (4) s.t. properties 1.-4. in Def. 14 are satisfied. This allows to show by case distinction using the definition of e_P above that we have for all $(a,p) \in A_{type(p)} \otimes P_K$: $(a,p) \in IN(L_{init}) \Leftrightarrow (a, e_P(p)) \in IN(L_{init'})$ and $(a,p) \in OUT(L_{init}) \Leftrightarrow (a, e_P(p)) \in OUT(L_{init'})$.

The opposite direction, where $L_{init'} = L_{init'_2} \circ_{(J,j_4,j_3)} L_{init'_1}$ is given with pushout (4), follows by symmetry.

Equivalence of KI and KI' in Theorem 4 intuitively means that the AHL-processes KI_1 and KI_2 with consistent instantiations can be considered to be independent, because composition in each order leads to equivalent processes.

Given the two basic AHL-processes KI_{Eat} and KI_{Enter} with the AHL-net morphisms $mp_{Eat} : KI_{Eat} \to AN_{House}$ and $mp_{Enter} : KI_{Enter} \to AN_{House}$, where mp_{Eat} is defined in Section 2 and mp_{Enter} can be analogously defined. Because the properties 1. and 2. in Theorem 4 are satisfied by K_{Eat} and K_{Enter} as well as by mp_{Eat} and mp_{Enter}, we get the equivalent processes $KI_{Eat/Enter} = KI_{Eat} \circ_{(I,i_1,i_2)} K_{Enter}$ in Fig. 6 and $KI_{Enter/Eat} = KI_{Enter} \circ_{(I,i_4,i_3)} KI_{Eat}$ in Fig. 7.

5 Conclusion and Related Work

In this paper we have presented main results of a line of research concerning the modeling and analysis of high-level net processes. Based on the notions of high-level net processes with initial markings in [6, 7] we have introduced high-level net processes with instantiations. As main results we have presented suitable conditions for the composition and independence of high-level net processes. We have shown under these conditions that the composition of two high-level net processes leads to a high-level net process and they can be composed in any order leading to equivalent processes. In this case the two high-level net processes are called independent.

In [11,12] the semantics of object Petri nets is defined by a suitable extension of low-level processes. Objects Petri nets are high-level nets with P/T-systems as tokens. A process for an object Petri net is given by a pair of processes, a high-level net process containing low-level processes of the corresponding P/T-systems. In contrast the approach presented in this paper extends the notion of high-level net processes for algebraic high-level nets. The token structure of an algebraic high-level net is defined in its data type part that is not restricted to P/T-systems and we also use rules as tokens. For this reason low-level processes of P/T-systems as tokens are not considered.

Our main result of independence of high-level net processes is inspired by the results of local Church-Rosser for graph resp. net transformation [10,13], where under suitable conditions transformation steps can be performed in any order leading to the same result. In [14] we have transferred these results, so that net transformations and token firing can be executed in arbitrary order provided that certain conditions are satisfied. For example in Section 2 the firing step $table_{45} \xrightarrow{start\ eating_5} table'_{45}$ (see Fig. 2) and the transformation step $(phi_3, table_{45}) \overset{rule_2}{\Longrightarrow} table_{345}$ (see Fig. 3) are independent of each other, so that each of this evolution steps can be postponed after the realization of the other yielding the same result, the P/T-system $table'_{345}$ in Fig. 4. Hence an interesting aspect of future work will be to investigate the correspondence between these different concepts of independence in more detail to gain further results for high-level net processes.

References

1. Goltz, U., Reisig, W.: The Non-sequential Behavior of Petri Nets. Information and Control **57**(2/3) (1983) 125–147
2. Rozenberg, G.: Behaviour of Elementary Net Systems. In: Petri Nets: Central Models and Their Properties, Advances in Petri Nets. Volume 254 of LNCS., Springer (1987) 60–94
3. Degano, P., Meseguer, J., Montanari, U.: Axiomatizing Net Computations and Processes. In: Proc. on Logic in Computer Science (LICS), IEEE Computer Society (1989) 175–185
4. Engelfriet, J.: Branching Processes of Petri Nets. Acta Informatica **28**(6) (1991) 575–591
5. Meseguer, J., Montanari, U., Sassone, V.: On the Semantics of Place/Transition Petri Nets. Mathematical Structures in Computer Science **7**(4) (1997) 359–397
6. Ehrig, H.: Behaviour and Instantiation of High-Level Petri Net Processes. Fundamenta Informaticae **65**(3) (2005) 211–247
7. Ehrig, H., Hoffmann, K., Padberg, J., Baldan, P., Heckel, R.: High-level net processes. In: Formal and Natural Computing. Volume 2300 of LNCS., Springer (2002) 191–219
8. Hoffmann, K., Mossakowski, T., Ehrig, H.: High-Level Nets with Nets and Rules as Tokens. In: Proc. Application and Theory of Petri Nets (ATPN). Volume 3536 of LNCS. Springer (2005) 268–288

9. Meseguer, J., Montanari, U.: Petri Nets Are Monoids. Information and Computation **88**(2) (1990) 105–155
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in TCS. Springer (2006)
11. Farwer, B., Köhler, M.: Mobile Object-Net Systems and their Processes. Fundam. Inform. **60**(1-4) (2004) 113–129
12. Köhler, M., Rölke, H.: Reference and Value Semantics Are Equivalent for Ordinary Object Petri Nets. In: Proc. Application and Theory of Petri Nets (ATPN). Volume 3536 of LNCS., Springer (2005) 309–328
13. Rozenberg, G.: Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations. World Scientific (1997)
14. Ehrig, H., Hoffmann, K., Padberg, J., Prange, U., Ermel, C.: Independence of net transformations and token firing in reconfigurable place/transition systems. In: Proc. Application and Theory of Petri Nets (ATPN). Volume 4546 of LNCS., Springer (2007) 104–123

A Appendix

A.1 Signature and Algebra for P/T-Systems and Rules as Tokens

Definition 15 (HLNR-System-SIG Signature and Algebra).
Given vocabularies T_0 and P_0, the signature HLNR-System-SIG is given by HLNR-System-SIG =

sorts: $Transitions, Places, Bool, System, Mor, Rules$
opns: tt, ff: $\to Bool$
$enabled : System \times Transitions \to Bool$
$fire : System \times Transitions \to System$
$applicable : Rules \times Mor \to Bool$
$transform : Rules \times Mor \to System$
$coproduct : System \times System \to System$
$pr_{phi} : System \to System$
$pr_{table} : System \to System$
$cod : Mor \to System$

and the HLNR-System-SIG-algebra A for P/T-systems and rules as tokens is given by

- $A_{Transitions} = T_0, A_{Places} = P_0, A_{Bool} = \{true, false\}$,
- A_{System} *the set of all P/T-systems over T_0 and P_0, i.e.*
  $A_{System} = \{PN | PN = (P, T, pre, post, M)$ *P/T-system*, $P \subseteq P_0, T \subseteq T_0\}$
  $\cup \{undef\}$,
- A_{Mor} *the set of all P/T-morphisms for A_{System}, i.e.*
  $A_{Mor} = \{f | f : PN \to PN'$ *P/T-morphism with* $PN, PN' \in A_{System}\}$,
- A_{Rules} *the set of all rules of P/T-systems, i.e.*
  $A_{Rules} = \{r | r = (L \overset{i_1}{\leftarrow} I \overset{i_2}{\rightarrow} R)$ *rule of P/T-systems with strict inclusions* $i_1, i_2\}$,
- $tt_A = true$, $ff_A = false$,

- $enabled_A : A_{System} \times T_0 \to \{true, false\}$ *for* $PN = (P, T, pre, post, M)$ *with*

$$enabled_A(PN, t) = \begin{cases} true & \textit{if } t \in T, pre(t) \leq M \\ false & \textit{else} \end{cases}$$

- $fire_A : A_{System} \times T_0 \to A_{System}$ *for* $PN = (P, T, pre, post, M)$ *with*

$$fire_A(PN, t) = \begin{cases} (P, T, pre, post, M \ominus pre(t) \oplus post(t)) & \\ & \textit{if } enabled_A(PN, t) = tt \\ undef & \textit{else} \end{cases}$$

- $applicable_A : A_{Rules} \times A_{Mor} \to \{true, false\}$ *with*

$$applicable_A(r, m) = \begin{cases} true & \textit{if } r \textit{ is applicable at match } m \\ false & \textit{else} \end{cases}$$

- $transform_A : A_{Rules} \times A_{Mor} \to A_{System}$ *with*

$$transform_A(r, m) = \begin{cases} H & \textit{if } applicable_A(r, m) \\ undef & \textit{else} \end{cases}$$

  where for $L \xrightarrow{m} G$ *and* $applicable_A(r, m) = true$ *we have a direct transformation* $G \overset{r}{\Longrightarrow} H$,
- $coproduct_A : A_{System} \times A_{System} \to A_{System}$ *the disjoint union (i.e. the two P/T-systems are combined without interaction) with*

$$\begin{aligned} coproduct_A(PN_1, PN_2) = {} & \underline{if}\ (PN_1 = undef \vee PN_2 = undef)\ \underline{then}\ undef \\ & \underline{else}\ ((P_1 \uplus P_2), (T_1 \uplus T_2), pre_3, post_3, M_1 \oplus M_2) \end{aligned}$$

  where $pre_3, post_3 : (T_1 \uplus T_2) \to (P_1 \uplus P_2)^{\oplus}$ *are defined by*

$$\begin{aligned} pre_3(t) = {} & \underline{if}\ t \in T_1\ \underline{then}\ pre_1(t)\ \underline{else}\ pre_2(t) \\ post_3(t) = {} & \underline{if}\ t \in T_1\ \underline{then}\ post_1(t)\ \underline{else}\ post_2(t) \end{aligned}$$

- $pr_{phi,A} : A_{System} \to A_{System}$ *with*

$$pr_{phi,A}(PN) = \begin{cases} phi_i & \textit{if } PN = coproduct_A(phi_i, PN') \\ undef & \textit{else} \end{cases}$$

  where $phi_i = (\{p_i\}, \{t_i\}, pre_i, post_i)$ *with* $pre_i(t_i) = post_i(t_i) = p_i$,
- $pr_{table,A} : A_{System} \to A_{System}$ *with*

$$pr_{table,A}(PN) = PN \setminus pr_{phi,A}(PN)$$

- $cod_A : A_{Mor} \to A_{System}$ *with* $cod_A\ (f : PN_1 \to PN_2) = PN_2$.

A.2 Flattening and Skeleton Construction

Definition 16 (Firing Behaviour of AHL-Nets). *A marking of an AHL-net AN is given by $M \in CP^{\oplus}$ where $CP = (A \otimes P) = \{(a,p) | a \in A_{type(p)}, p \in P\}$.*

The set of variables $Var(t) \subseteq X$ of a transition $t \in T$ are the variables of the net inscriptions in $pre(t), post(t)$ and $cond(t)$. Let $v : Var(t) \to A$ be a variable assignment with term evaluation $v^{\sharp} : T_{\Sigma}(Var(t)) \to A$, then (t,v) is a consistent transition assignment iff $cond_{AN}(t)$ is validated in A under v. The set CT of consistent transition assignments is defined by $CT = \{(t,v) | (t,v)$ consistent transition assignment$\}$.

A transition $t \in T$ is enabled in M under v iff $(t,v) \in CT$ and $pre_A(t,v) \leq M$, where $pre_A : CT \to CP^{\oplus}$ defined by $pre_A(t,v) = \hat{v}(pre(t)) \in (A \otimes P)^{\oplus}$ and $\hat{v} : (T_{\Sigma}(Var(t)) \otimes P)^{\oplus} \to (A \otimes P)^{\oplus}$ is the obvious extension of $v^{\sharp}$ to terms and places (similar $post_A : CT \to CP^{\oplus}$). Then the follower marking is computed by $M' = M \ominus pre_A(t,v) \oplus post_A(t,v)$.

Definition 17 (Flattening). *Given AHL-net AN as above then the flattening of AN is a P/T-net $Flat(AN) = N = (CP, CT, pre_A, post_A)$ with*

- *$CP = A \otimes P = \{(a,p) | a \in A_{type(p)}, p \in P\}$,*
- *$CT = \{(t,v) | t \in T, v : Var(t) \to A$ s.t. $cond(t)$ valid in A under v $\}$ and*
- *pre_A and $post_A$ as defined in Def. 16.*

Given an AHL-net morphism $f : AN_1 \to AN_2$ by $f = (f_P, f_T)$ then $Flat(f) = (id_A \otimes f_P : CP_1 \to CP_2, f_C : CT_1 \to CT_2)$ is given by $id_A \otimes f_P(a,p) = (a, f_p(p))$ and $f_C(t,v) = (f_T(t), v)$.

Definition 18 (Skeleton). *Given an AHL-net AN as above then the skeleton of AN is a P/T-net $Skel(AN) = (P, T, pre_S, post_S)$ with $pre_S(t) = \sum_{i=1}^{n} p_i$ for $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ and similar for $post_S : T \to P^{\oplus}$. Given an AHL-net morphism $f : AN_1 \to AN_2$ by $f = (f_P, f_T)$ then $Skel(f) = f = (f_P : P_1 \to P_2, f_T : T_1 \to T_2)$.*

Remark 1. The flattening construction defined in Def. 17 and the skeleton construction defined in Def. 18 are well-defined and can be turned into a functor $Flat$: **AHLNet** → **PTNet** and a functor $Skel$: **AHLNet** → **PTNet** which preserve pushouts, i.e. given the pushout (1) in **AHLNet** then there are corresponding pushouts (2) and (3) in **PTNet**. Moreover we have for each AN a projection $proj(AN) : Flat(AN) \to Skel(AN)$ leading to a natural transformation $proj : Flat \to Skel$.

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

$$\begin{tikzcd}
Flat(AN_0) \arrow[r, "Flat(f_1)"] \arrow[d, "Flat(f_2)"'] \arrow[dr, phantom, "(\mathbf{2})"] & Flat(AN_1) \arrow[d, "Flat(f_1')"] \\
Flat(AN_2) \arrow[r, "Flat(f_2')"'] & Flat(AN_3)
\end{tikzcd}
\qquad
\begin{tikzcd}
Skel(AN_0) \arrow[r, "Skel(f_1)"] \arrow[d, "Skel(f_2)"'] \arrow[dr, phantom, "(\mathbf{3})"] & Skel(AN_1) \arrow[d, "Skel(f_1')"] \\
Skel(AN_2) \arrow[r, "Skel(f_2')"'] & Skel(AN_3)
\end{tikzcd}$$

Theorem 5 (*Flat* is functor). *The construction* $Flat : \mathbf{AHLNet} \to \mathbf{PTNet}$ *as defined in Def. 17 is a functor.*

Proof.

1. $Flat(AN)$ is well-defined:
   We have to show that $pre_A, post_A \in CP^{\oplus}$. This follows for pre_A from the fact that $term_i$ of $type(p_i)$ implies $v^{\sharp}(term_i) \in A_{type(p_i)}$ and similar for $post_A$.
2. $Flat(f)$ is well-defined:
   (a) Let $(a, p) \in CP_1$.
   By definition we have $Flat(f)_P(a, p) = (id_A \otimes f_P)(a, p) = (id_A(a), f_P(p)) = (a, f_P(p)) \in CP_2$ because $f_P(p) \in P_2$ and $a \in A_{type_1(p)} = A_{type_2(f_P(p))}$.
   (b) $(t, v) \in CT_1 \Rightarrow f_C(t, v) \in CT_2$
   $(t, v) \in CT_1$ means $v : Var(t) \to A$ s.t. $cond_1(t)$ valid in A under v,
   $f_C(t, v) = (f_T(t), v) \in CT_2$ means $v : Var(f_T(t)) \to A$ s.t. $cond_2(f_T(t)) = cond_1(t)$ valid in A under v. This follows from $(t, v) \in CT_1$ because

$$\begin{aligned}
Var(t) &= Var(cond_1(t)) \cup Var(pre_1(t)) \cup Var(post_1(t)) \\
&= Var(cond_2(f_T(t))) \cup Var(pre_2(f_T(t))) \cup Var(post_2(f_T(t))) \\
&= Var(f_T(t))
\end{aligned}$$

3. $Flat(f)$ is P/T-net morphism:
   For symmetry reasons it suffices to show commutativity for pre_A of

$$\begin{tikzcd}
CT_1 \arrow[r, "pre_{1,A}"] \arrow[d, "f_C"'] \arrow[dr, phantom, "(\mathbf{1})"] & CP_1^{\oplus} \arrow[d, "(id_A \otimes f_P)^{\oplus}"] \\
CT_2 \arrow[r, "pre_{2,A}"'] & CP_2^{\oplus}
\end{tikzcd}$$

Given $(t_1, v_1) \in CT_1$ with $pre_1(t_1) = \sum_{i=1}^{n}(term_i, p_i)$ we have
$pre_2(f_T(t_1)) = (id_{T_{\Sigma}(X)} \otimes f_P)^{\oplus}(pre_1(t_1)) = (id_{T_{\Sigma}(X)} \otimes f_P)^{\oplus}(\sum_{i=1}^{n}(term_i, p_i))$
$= \sum_{i=1}^{n}(term_i, f_P(p_i))$
because f is an AHL-morphisms and hence
$(id_A \otimes f_P)^{\oplus}(pre_{1A}(t_1, v_1)) = (id_A \otimes f_P)^{\oplus}(\sum_{i=1}^{n}(v_1^{\sharp}(term_i), p_i))$
$= \sum_{i=1}^{n}(v_1^{\sharp}(term_i), f_P(p_i)) = pre_{2A}(f_T(t_1), v_1) = pre_{2A}(f_C(t_1, v_1))$.

4. Obviously we have $Flat(f) = id_{Flat(AN)}$ for $f = id_{AN}$. Furthermore we have for the composition $Flat(g \circ f) = ((id_A \otimes (g \circ f)_P)^{\oplus}, (g \circ f)_C) = ((id_A \otimes (g \circ f)_P)^{\oplus}, g_C \circ f_C) = ((id_A \circ id_A \otimes (g \circ f)_P)^{\oplus}, g_C \circ f_C) = ((id_A \otimes g_P)^{\oplus} \circ (id_A \otimes f_P)^{\oplus}, g_C \circ f_C) = Flat(g) \circ Flat(f)$.

Theorem 6 (*Flat* preserves Pushouts). *Given Pushout* (1) *in* **AHLNet** *then* (2) *is Pushout in* **PTNet**.

$$\begin{tikzcd}
AN_0 \arrow[r, "f_1"] \arrow[d, "f_2"'] \arrow[dr, phantom, "(\mathbf{1})"] & AN_1 \arrow[d, "g_1"] \\
AN_2 \arrow[r, "g_2"'] & AN_3
\end{tikzcd}
\qquad
\begin{tikzcd}
Flat(AN_0) \arrow[r, "Flat(f_1)"] \arrow[d, "Flat(f_2)"'] \arrow[dr, phantom, "(\mathbf{2})"] & Flat(AN_1) \arrow[d, "Flat(g_1)"] \\
Flat(AN_2) \arrow[r, "Flat(g_2)"'] & Flat(AN_3)
\end{tikzcd}$$

Proof. Pushout (1) in **AHLNet** is constructed componentwise by the Pushouts

$$\begin{tikzcd}
T_0 \arrow[r, "f_{1,T}"] \arrow[d, "f_{2,T}"'] \arrow[dr, phantom, "(\mathbf{3})"] & T_1 \arrow[d, "g_{1,T}"] \\
T_2 \arrow[r, "g_{2,T}"'] & T_3
\end{tikzcd}
\qquad
\begin{tikzcd}
P_0 \arrow[r, "f_{1,P}"] \arrow[d, "f_{2,P}"'] \arrow[dr, phantom, "(\mathbf{4})"] & P_1 \arrow[d, "g_{1,P}"] \\
P_2 \arrow[r, "g_{2,P}"'] & P_3
\end{tikzcd}$$

in **SET**. Since Pushouts in **PTNet** are also constructed componentwise in **SET** it suffices to show that (5) and (6) are Pushouts in **SET**.

$$\begin{tikzcd}
CT_0 \arrow[r, "f_{1,C}"] \arrow[d, "f_{2,C}"'] \arrow[dr, phantom, "(\mathbf{5})"] & CT_1 \arrow[d, "g_{1,C}"] \\
CT_2 \arrow[r, "g_{2,C}"'] & CT_3
\end{tikzcd}
\qquad
\begin{tikzcd}
A \otimes P_0 \arrow[r, "id_A \otimes f_{1,P}"] \arrow[d, "id_A \otimes f_{2,P}"] \arrow[dr, phantom, "(\mathbf{6})"] & A \otimes P_1 \arrow[d, "id_A \otimes g_{1,P}"'] \arrow[ddr, bend left, "h_1"] & \\
A \otimes P_2 \arrow[r, "id_A \otimes g_{2,P}"] \arrow[drr, bend right, "h_2"] & A \otimes P_3 \arrow[r, phantom, "(\mathbf{7})"] \arrow[d, phantom, "(\mathbf{8})"] \arrow[dr, dotted, "h"] & {} \\
 & {} & X
\end{tikzcd}$$

We show the universal properties for (6), because we cannot directly use that the cartesian product $A \times \_$ preserves Pushouts, since $A \otimes P \subsetneq A \times P$. Given h_1, h_2 with $h_1 \circ (id_A \otimes f_{1P}) = h_2 \circ (id_A \otimes f_{2P})$ we define $h : A \otimes P_3 \to X$ by

$$h(a, p_3) = \begin{cases} h_1(a, p_1) & \text{for } p_3 = g_{1P}(p_1) \text{ with } p_1 \in P_1 \\ h_2(a, p_2) & \text{for } p_3 = g_{2P}(p_2) \text{ with } p_2 \in P_2 \end{cases}$$

It suffices to show that h is well-defined, i.e. $p_3 = g_{1P}(p_1) = g_{2P}(p_2)$ implies $h_1(a, p_1) = h_2(a, p_2)$, because with this definition (7) and (8) commute by construction and h is unique with this property.

Given $p_3 = g_{1P}(p_1) = g_{2P}(p_2)$ we have by definition of Pushout (4) in **SET** a sequence $p_{01}, \dots, p_{0n} \in P_0$ with

$$\begin{tikzcd}
 & p_{01} \arrow[dl, mapsto, "f_{1,P}"'] \arrow[dr, mapsto, "f_{2,P}"] & & p_{02} \arrow[dl, mapsto, "f_{2,P}"'] \arrow[dr, mapsto, "f_{1,P}"] & & p_{03} \arrow[dl, mapsto, "f_{1,P}"'] & \cdots & p_{0n} \arrow[dr, mapsto, "f_{2,P}"] & \\
p_1 & & p_{21} & & p_{11} & & \cdots & & p_2
\end{tikzcd}$$

This implies

$$\begin{aligned}
h_1(a, p_1) &= h_1(a, f_{1P}(p_{01})) \\
&= h_1 \circ (id_A \otimes f_{1P})(a, p_{01}) \\
&= h_2 \circ (id_A \otimes f_{2P})(a, p_{01}) \\
&= h_2(a, f_{2P}(p_{01})) \\
&= h_2(a, p_{21}) \\
&= h_2(a, f_{2P}(p_{02})) \\
&= h_2 \circ (id_A \otimes f_{2P})(a, p_{02}) \\
&= h_1 \circ (id_A \otimes f_{1P})(a, p_{02}) \\
&= \dots \\
&= h_1 \circ (id_A \otimes f_{1P})(a, p_{03}) \\
&= \dots \\
&= h_2 \circ (id_A \otimes f_{2P})(a, p_{0n}) \\
&= h_2(a, f_{2P}(p_{0n})) \\
&= h_2(a, p_2)
\end{aligned}$$

All steps are well-defined, because $(a, p_{01}), \dots, (a, p_{0n}) \in A \otimes P_0$:
In fact $(a, p_3) \in A \otimes P_3$ implies $a \in A_{type_3(p_3)} = A_{type_1(p_1)} = A_{type_0(p_{0i})}$ for $i = 1, \dots, n$ using type compability of g_{1P} for $p_3 = g_{1P}(p_1)$ and of f_{1P}, f_{2P}, g_{2P} for the other cases (see proof of Theorem 5).

For similar reasons we can show explicitely the universal properties of (5) using Pushout (3). Given $k_1 : CT_1 \to X, k_2 : CT_2 \to X$ with $k_1 \circ f_{1C} = k_2 \circ f_{2C}$ there is a unique $k : CT_3 \to X$ defined by

$$k(t_3, v) = \begin{cases} k_1(t_1, v) & \text{for } t_3 = g_{1T}(t_1) \text{ with } t_1 \in T_1 \\ k_2(t_2, v) & \text{for } t_3 = g_{2T}(t_2) \text{ with } t_2 \in T_2 \end{cases}$$

Similar to above we can show that $t_3 = g_{1T}(t_1) = g_{2T}(t_2)$ implies $k_1(t_1, v) = k_2(t_2, v)$ where all steps are well-defined using Theorem 5.

Theorem 7 (*Flat* preserves Monomorphisms). *Given $f : AN_1 \to AN_2$ monomorphism in* **AHLNet***, then $Flat(f) : Flat(AN_1) \to Flat(AN_2)$ is monomorphism in* **PTNet***.*

Proof. Since monomorphisms in **AHLNet** and **PTNet** are the injective morphisms, it suffices to show: If f is injective, $Flat(f)$ is injective.

Let us assume that f is injective. That means f_P and f_T are injective functions.
Let $Flat(f)_P(a_1, p_1) = Flat(f)_P(a_2, p_2)$. By Def. 17 we have:
$Flat(f)_P(a_1, p_1) = Flat(f)_P(a_2, p_2)$
$\Rightarrow (id_A \otimes f_P)(a_1, p_1) = (id_A \otimes f_P)(a_2, p_2)$
$\Rightarrow (a_1, f_P(p_1)) = (a_2, f_P(p_2))$
$\Rightarrow a_1 = a_2 \wedge f_P(p_1) = f_P(p_2)$
$\Rightarrow a_1 = a_2 \wedge p_1 = p_2$ because f_P is injective
$\Rightarrow (a_1, p_1) = (a_2, p_2)$
$\Rightarrow Flat(f)_P$ is injective.

Let $Flat(f)_T(t_1, v_1) = Flat(f)_T(t_2, v_2)$. By Def. 17 we have:
$Flat(f)_T(t_1, v_1) = Flat(f)_T(t_2, v_2)$
$\Rightarrow (f_T(t_1), v_1) = (f_T(t_2), v_2)$
$\Rightarrow f_T(t_1) = f_T(t_2) \wedge v_1 = v_2$
$\Rightarrow t_1 = t_2 \wedge v_1 = v_2$ because f_T is injective
$\Rightarrow (t_1, v_1) = (t_2, v_2)$
$\Rightarrow Flat(f)_T$ is injective.
$Flat(f)_P$ and $Flat(f)_T$ are injective implies that $Flat(f)$ is injective.

Theorem 8 **(*Skel* is functor).** *The construction* $Skel$: **AHLNet** $\to$ **PTNet** *as defined in Def. 18 is a functor.*

Proof. Given $f : AN_1 \to AN_2$ then $Skel(f) : Skel(AN_1) \to Skel(AN_2)$ is P/T-net morphism, i.e. (1) commutes componentwise.

$$\begin{tikzcd}
T_1 \arrow[r, shift left, "pre_{1,S}"] \arrow[r, shift right, "post_{1,S}"'] \arrow[d, "f_T"'] \arrow[dr, phantom, "(\mathbf{1})"] & P_1^{\oplus} \arrow[d, "f_P^{\oplus}"] \\
T_2 \arrow[r, shift left, "pre_{2,S}"] \arrow[r, shift right, "post_{2,S}"'] & P_2^{\oplus}
\end{tikzcd}$$

For $t_1 \in T_1$ and $pre_1(t_1) = \sum_{i=1}^{n}(term_i, p_i)$ we have
$f_P \circ pre_{1S}(t_1) = f_P^{\oplus}(\sum_{i=1}^{n} p_i) = \sum_{i=1}^{n} f_P(p_i)$
$pre_2 \circ f_T(t_1) = (id_A \otimes f_P)^{\oplus} \circ pre_1(t_1) = \sum_{i=1}^{n}(term_i, f_P(p_i))$ and hence
$pre_{2S} \circ f_T(t_1) = pre_{2S}(f_T(t_1)) = \sum_{i=1}^{n} f_P(p_i)$ which implies
$f_P \circ pre_{1S}(t_1) = pre_{2S} \circ f_T(t_1)$ and similar for $post_{1S}$ and $post_{2S}$.

Theorem 9 **(*Skel* preserves Pushouts).** *Given Pushout* (1) *in* **AHLNet** *the* (2) *is Pushout in* **PTNet**.

$$\begin{tikzcd}
AN_0 \arrow[r, "f_1"] \arrow[d, "f_2"'] \arrow[dr, phantom, "(\mathbf{1})"] & AN_1 \arrow[d, "g_1"] \\
AN_2 \arrow[r, "g_2"'] & AN_3
\end{tikzcd}
\qquad
\begin{tikzcd}
Skel(AN_0) \arrow[r, "Skel(f_1)"] \arrow[d, "Skel(f_2)"'] \arrow[dr, phantom, "(\mathbf{2})"] & Skel(AN_1) \arrow[d, "Skel(g_1)"] \\
Skel(AN_2) \arrow[r, "Skel(g_2)"'] & Skel(AN_3)
\end{tikzcd}$$

Proof. Pushout (1) in **AHLNet** implies Pushouts (3) and (4) as in Theorem 6. But this implies that (2) is Pushout in **PTNet** , because Pushouts in **PTNet** are based on Pushouts (3) and (4) in **SET**.

Theorem 10 (*Skel* preserves Monomorphisms). *Given $f : AN_1 \to AN_2$ monomorphism in* **AHLNet***, then $Skel(f) : Skel(AN_1) \to Skel(AN_2)$ is monomorphism in* **PTNet***.*

Proof. Similar to the proof that $Flat$ preserves monomorphisms, we will show: If f is injective, $Skel(f)$ is injective.

Let us assume that $f : AN_1 \to AN_2$ is injective. That means f_P and f_T are injective functions.
Let $Skel(f)_P(p_1) = Skel(f)_P(p_2)$ for $p_1, p_2 \in P_{Skel(AN_1)}$. By Def. 18 we have:
$Skel(f)_P(p_1) = Skel(f)_P(p_2)$
$\Rightarrow f_P(p_1) = f_P(p_2)$
$\Rightarrow p_1 = p_2$ because f_P is injective. Hence $Skel(f)_P$ is injective.

Let $Skel(f)_T(t_1) = Skel(f)_T(t_2)$ for $t_1, t_2 \in T_{Skel(AN_1)}$. By Def. 18 we have:
$Skel(f)_T(t_1) = Skel(f)_T(t_2)$
$\Rightarrow f_T(t_1) = f_T(t_2)$
$\Rightarrow t_1 = t_2$ because f_T is injective. Hence $Skel(f)_T$ is injective.

A.3 Proof of Theorem 1 in Section 4

Proof. We have to show

1. Unarity
2. No forward conflicts
3. No backward conflicts
4. The causal relation is a finitary strict partial order.

1. Unarity:

   Let us assume that K does not comply with the unarity property, i.e. there is $t \in T$ and $(term_1, p), (term_2, p) \in T_\Sigma(X) \otimes P$ s.t. $(term_1, p) \oplus (term_2, p) \leq pre(t)$ or $(term_1, p) \oplus (term_2, p) \leq post(t)$.
   The set of transitions T is the Pushout $T_1 \overset{i_{1T}}{\leftarrow} \emptyset \overset{i_{2T}}{\rightarrow} T_2$ which is the disjoint union of T_1 and T_2, and hence there is $t_x \in T_x$ with $t = i'_{x,T}(t_x)$ where either $x = 1$ or $x = 2$.
   Since i'_x is an AHL-morphism it preserves pre and post domains, i.e. $pre \circ$

$i'_{x,T}(t_x) = (id_{T_\Sigma(X)} \otimes i'_{x,P})^{\oplus} \circ pre_x(t_x)$.
This implies for $(term_1, p) \oplus (term_2, p) \leq pre(t) = pre(i_{x,T}(t_x))$ that $(term_1, p) \oplus (term_2, p) \leq (id_{T_\Sigma(X)} \otimes i'_{x,P})^{\oplus} \circ pre_x(t_x)$, which means that there are $(term_1, p_1)$, $(term_2, p_2) \in T_\Sigma(X) \otimes P_x$ s.t. $(term_1, p_1) \oplus (term_2, p_2) \leq pre_x(t_x)$ and $i'_{x,P}(p_1) = p = i'_{x,P}(p_2)$.
The fact that $i'_{x,P}(p_1) = i'_{x,P}(p_2)$ implies that $p_1 = p_2$ because $i'_{x,P}$ is injective and hence $(term_1, p_1) \oplus (term_2, p_1) \leq pre_x(t_x)$ contradicts the fact that K_x complies with the unarity property.

The case $(term_1, p) \oplus (term_2, p) \leq post(t)$ works analogously.

2. No forward conflict:

   We have to show:
   $$\forall t, t' \in T : t \neq t' \Rightarrow \bullet t \cap \bullet t' = \emptyset$$

   Let us assume that there is a forward conflict, i.e. there exist $t, t' \in T, p \in P$, s.t. $t \neq t'$ and $p \in \bullet t, p \in \bullet t'$.

   Case 1: Both transitions t and t' have a preimage in the same net, i.e. there are $t_x, t'_x \in T_x$ with $i'_{xT}(t_x) = t$ and $i'_{xT}(t'_x) = t'$ where either $x = 1$ or $x = 2$. Due to $i'_{xT}(t_x) \neq i'_{xT}(t'_x)$ we also have $t_x \neq t'_x$.
   $p \in \bullet t$ and $p \in \bullet t'$ implies that there exist $p_x, p'_x \in P_x$, s.t. $i'_{xP}(p_x) = i'_{xP}(p'_x) = p$ and $p_x \in \bullet t_x, p'_x \in \bullet t'_x$, because i'_x is an AHL-morphism which preserves pre and post domains. Since i'_{xP} is injective $i'_{xP}(p_x) = i'_{xP}(p'_x)$ implies $p_x = p'_x$ and hence $p_x \in \bullet t_x$ and $p_x \in \bullet t'_x$ which contradicts the fact that K_x has no forward conflict.

   Case 2: The transitions t, t' have their preimage in different nets, i.e there exist $t_1 \in T_1, t_2 \in T_2$ with $t = i'_{1T}(t_1), t' = i'_{2T}(t_2)$. Since i'_1 and i'_2 preserve pre and post domains, there exist $p_1 \in P_1, p_2 \in P_2$ with $p_1 \in \bullet t_1$, $p_2 \in \bullet t_2$ and $i'_{1P}(p_1) = i'_{2P}(p_2)$. This means that there is a $p_0 \in P_I$ with $i_{1P}(p_0) = p_1$ and $i_{2P}(p_0) = p_2$, because K is the Pushout of $K_1 \overset{i_1}{\leftarrow} I \overset{i_2}{\rightarrow} K_2$. The fact that K_1 and K_2 are composable wrt. (I, i_1, i_2) implies for $p_1 \in i_1(P_I)$ that $p_1 \in OUT(K_1)$ which contradicts $p_1 \in \bullet t_1$.

3. No backward conflict:

   We have to show:
   $$\forall t, t' \in T : t \neq t' \Rightarrow t \bullet \cap t' \bullet = \emptyset$$

   Let us assume that there is a backward conflict, i.e. there exist $t, t' \in T, p \in P$, s.t. $p \in t\bullet$ and $p \in t'\bullet$.
   Case 1: Both transitions t and t' have a preimage in the same net, i.e. there are $t_x, t'_x \in T_x$ with $i'_{xT}(t_x) = t$ and $i'_{xT}(t'_x) = t'$ where either $x = 1$ or $x = 2$. Analogously to case 1 of the forward conflicts it follows that there exist

$p_x \in P_x$, s.t. $p_x \in t_x\bullet$ and $p_x \in t'_x\bullet$. This contradicts the fact that K_x has no backward conflict.
Case 2: The transitions t, t' have their preimage in different nets, i.e there exist $t_1 \in T_1, t_2 \in T_2$ with $t = i'_{1T}(t_1), t' = i'_{2T}(t_2)$. Analogously to case 2 of the forward conflicts there exist $p_1 \in P_1, p_2 \in P_2$ with $p_1 \in t_1\bullet$ and $p_2 \in t_2\bullet$ where $i'_{1P}(p_1) = i'_{2P}(p_2)$. Due to the fact that K is the Pushout of $K_1 \xleftarrow{i_1} I \xrightarrow{i_2} K_2$ there is a $p_0 \in P_I$ with $i_{1P}(p_0) = p_1$ und $i_{2P}(p_0) = p_2$, which implies for $p_2 \in i_{2P}(P_I)$ that $p_2 \in IN(K_2)$, because K_1 and K_2 are composable wrt. (I, i_1, i_2). This contradicts $p_2 \in t_1\bullet$.

4. The causal relation is a finitary strict partial order:

   The composability of K_1 and K_2 wrt. (I, i_1, i_2) requires that $i_1(P_I) \subseteq OUT(K_1)$ and $i_2(P_I) \subseteq IN(K_2)$. That means that exclusively output places (i.e. places without post domain) of K_1 with input places (i.e. places without pre domain) of K_2 are identified.
   Due to the fact that AHL-morphisms preserve pre and post domains of transitions, we obtain the causal relation $<_K$ as the transitive closure of
   $\{(i'_1(x), i'_1(y))|x, y \in P_{K_1} \uplus T_{K_1}, x <_{K_1} y\} \cup \{(i'_2(x), i'_2(y))|x, y \in P_{K_2} \uplus T_{K_2}, x <_{K_2} y\}$.
   This means that the causal relations of the nets K_1 and K_2 are also causal relations in K and additionally it is possible that nodes $x_2 \in P_{K_2} \uplus T_{K_2}$ originated from K_2 can be successor of nodes $x_1 \in P_{K_1} \uplus T_{K_1}$ originated from K_1 (i.e. $i'_1(x_1) <_K i'_2(x_2)$), but not vice versa (i.e. $i'_2(x_2) \not<_K i'_1(x_1)$).

   Let us assume that $<_K$ is not finitary, i.e. there exists $x \in P_K \uplus T_K$ with an infinite number of predecessors. That means that there are infinite $x_n <_K x$ $(n \in \mathbb{N})$.
   Case 1: Let $x = i'_1(x')$ with $x' \in P_{K_1} \uplus T_{K_1}$. Then we have $x'_n \in P_{K_1} \uplus T_{K_1}$ $(n \in \mathbb{N})$, s.t. for all $n \in \mathbb{N} : x'_n <_{K_1} x'$. This contradicts the fact that $<_{K_1}$ is finitary.
   Case 2: Let $x = i'_2(x'')$ with $x'' \in P_{K_2} \uplus T_{K_2}$. Let us assume that there exist infinite $x''_n \in P_{K_2} \uplus T_{K_2}$ $(n \in \mathbb{N})$, s.t. for all $n \in \mathbb{N} : x''_n <_{K_2} x''$. This contradicts the fact that $<_{K_2}$ is finitary.
   Hence there exists $m \in \mathbb{N}$, s.t. for $0 \leq k \leq m : x''_k <_{K_2} x''$ and $x''_m \in IN(K_2)$. Due to the fact that $i'_2(x''_m)$ has an infinite number of predecessors in $<_K$, there exists $y \in P_I$, s.t. $i'_2(x''_m) = i'_2(i_2(y)) = i'_1(i_1(y))$ has an infinite number of predecessors. This is equal to case 1.
   $<_K$ is finitary.

   Let us assume that $<_K$ is not irreflexive, i.e. there exists $x \in P_K \uplus T_K$, s.t. $x <_K x$. This means that there is a cycle in K and hence there exists $x' \in P_K \uplus T_K$, s.t. $x <_K x'$ and $x' <_K x$.
   Case 1: Both nodes x and x' are derived from the same net, i.e. for either $n = 1$ or $n = 2$ there exist $x_n, x'_n \in P_{K_n} \uplus T_{K_n}$, s.t. $x = i'_n(x_n)$ and $x' = i'_n(x'_n)$. Then we have $i'_n(x_n) <_K i'_n(x'_n)$ and $i'_n(x'_n) <_K i'_n(x_n)$, which

implies $x_n <_{K_n} x'_n$ and $x'_n <_{K_n} x_n$. This contradicts the fact that $<_{K_n}$ is irreflexive.
Case 2: The nodes x and x' are derived from different nets, i.e. there exist $x_1 \in P_{K_1} \uplus T_{K_1}, x'_2 \in P_{K_2} \uplus T_{K_2}$ with $x = i'_1(x_1)$ and $x' = i'_2(x'_2)$, s.t. $i'_1(x_1) <_K i'_2(x'_2)$ and $i'_2(x'_2) <_K i'_1(x_1)$. This contradicts the fact that according to the construction it is not possible that there exist $a \in P_{K_1} \uplus T_{K_1}$ and $b \in P_{K_2} \uplus T_{K_2}$, s.t. $i'_2(b) <_K i'_1(a)$.
Hence $<_K$ is irreflexive.

A.4 Proof of Theorem 2 in Section 4

Theorem 11 (Natural Transformation $proj : Flat \to Skel$).
$proj : Flat \to Skel$ defined for AHL-nets AN by
$proj(AN) : Flat(AN) \to Skel(AN)$ with
$proj(AN)_P(a, p) = p$ for $(a, p) \in CP = A \otimes P$ and
$proj(AN)_T(t, v) = t$ for $(t, v) \in CT$
is a natural transformation.

Proof. First we show that $proj(AN)$ is P/T-net morphism

$$\begin{tikzcd}
Flat(AN) \arrow[d, "proj(AN)"'] & & CT \arrow[r, shift left, "pre_A"] \arrow[r, shift right, "post_A"'] \arrow[d, "proj(AN)_T"'] \arrow[dr, phantom, "(\mathbf{1})"] & CP^{\oplus} \arrow[d, "proj(AN)_P^{\oplus}"] \\
Skel(AN) & & T \arrow[r, shift left, "pre_S"] \arrow[r, shift right, "post_S"'] & P^{\oplus}
\end{tikzcd}$$

Given $(t, v) \in CT$ with $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ we have
$proj(AN)_P^{\oplus} \circ pre_A(t, v) = proj(AN)_P^{\oplus}(\sum_{i=1}^{n}(v^{\sharp}(term_i), p_i)) = \sum_{i=1}^{n} p_i$
$pre_S \circ proj(AN)_T(t, v) = pre_S(t) = \sum_{i=1}^{n} p_i$
and similar for $post_A, post_S$.

In order to show that $proj$ is natural transformation let $f : AN_1 \to AN_2$ be an AHL-net morphism then we have to show commutativity of

$$\begin{tikzcd}
Flat(AN_1) \arrow[r, "Flat(f)=(id_A \otimes f_P, f_C)"] \arrow[d, "proj(AN_1)"'] \arrow[dr, phantom, "(\mathbf{2})"] & Flat(AN_2) \arrow[d, "proj(AN_2)"] \\
Skel(AN_1) \arrow[r, "Skel(f)=(f_P, f_T)"'] & Skel(AN_2)
\end{tikzcd}$$

This follows for $(a, p) \in A \otimes P_1$ from
$proj(AN_2)_P \circ (id_A \otimes f_P)(a, p) = f_P(p) = f_P \circ proj(AN_1)_P(a, p)$
and for $(t, v) \in CT_1$ from
$proj(AN_2)_T \circ f_C(t, v) = proj(AN_2)_T(f_T(t), v) = f_T(t) = f_T \circ proj(AN_1)_T(t, v)$.

Lemma 1 (Instantiation interface is Pullback).

The induced instantiation interface (J, j_1, j_2) of (L_{init_1}, L_{init_2}) as defined in Definition 11 in Section 4 is Pullback of $L_{init_1} \xrightarrow{in_1} Flat(K_1) \xleftarrow{Flat(i_1)} Flat(I)$, i.e. the diagram (1) is Pullback in **PTNet** *where in_1, in_I are inclusions.*

$$\begin{tikzcd}
& L_{init_1} \arrow[rr, "in_1"] \arrow[dr, phantom, "\mathbf{(1)}"] & & Flat(K_1) \\
J \arrow[ur, "j_1"] \arrow[rr, "in_I"'] & & Flat(I) \arrow[ur, "Flat(i_1)"'] &
\end{tikzcd}$$

Proof (Lemma 1).

We will show that the places and transitions are Pullbacks, i.e. (2) and (3) are Pullbacks in **SET**.

$$\begin{tikzcd}
& T_{L_{init_1}} \arrow[rr, "in_{1,T}"] \arrow[dr, phantom, "\mathbf{(2)}"] & & T_{Flat(K_1)} \\
T_J \arrow[ur, "j_{1,T}"] \arrow[rr, "in_{I,T}"'] & & T_{Flat(I)} \arrow[ur, "Flat(i_1)_T"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& P_{L_{init_1}} \arrow[rr, "in_{1,P}"] \arrow[dr, phantom, "\mathbf{(3)}"] & & P_{Flat(K_1)} \\
P_J \arrow[ur, "j_{1,P}"] \arrow[rr, "in_{I,P}"'] & & P_{Flat(I)} \arrow[ur, "Flat(i_1)_P"'] &
\end{tikzcd}$$

1. Pullback (2)
   Since $T_I = \emptyset$ also $T_{Flat(I)} = \emptyset$ and hence the Pullback of $T_{L_{init_1}} \xrightarrow{in_{1,T}} T_{Flat(K_1)} \xleftarrow{Flat(i_1)_T} T_{Flat(I)}$ is $\emptyset = T_J$.
2. Pullback (3)
   (a) Commutativity of (3):
   For $(a, p) \in P_J$ we have
   $in_{1P} \circ j_{1P}(a, p) = in_{1P} \circ (id_A \otimes i_{1P})(a, p) = in_{1P}(a, i_{1P}(p)) = (a, i_{1P}(p)) = Flat(i_1)_P(a, p) = Flat(i_1)_P \circ in_{IP}(a, p)$
   (b) Universal property:
   Given (Y, k_1, k_2) with $in_{1P} \circ k_1 = Flat(i_1)_P \circ k_2$.

$$\begin{tikzcd}
& & P_{L_{init_1}} \arrow[rr, "in_{1,P}"] \arrow[dr, phantom, "\mathbf{(3)}"] & & P_{Flat(K_1)} \\
& P_J \arrow[ur, "j_{1,P}"] \arrow[rr, "in_{I,P}"'] \arrow[l, phantom, "\mathbf{(4)}"] \arrow[d, phantom, "\mathbf{(5)}"] & & P_{Flat(I)} \arrow[ur, "Flat(i_1)_P"'] & \\
Y \arrow[uurr, bend left, "k_1"] \arrow[ur, dotted, "k"] \arrow[urrr, bend right, "k_2"'] & & & &
\end{tikzcd}$$

We define $k := k_2$. First we have to show that for all $y \in Y : k_2(y) \in P_J$.
Let $y \in Y$. Then we have $k_2(y) = (a,p) \in P_{Flat(I)}$ and hence $Flat(i_1)_P(k_2(y)) = (a, i_{1P}(p)) \in P_{Flat(K_1)}$.
$in_{1P} \circ k_1 = Flat(i_1)_P \circ k_2$ implies $in_{1P}(k_1(y)) = Flat(i_1)_P(k_2(y))$ and since in_{1P} is an inclusion, we have $k_1(y) = Flat(i_1)_P(k_2(y)) = (a, i_{1P}(p)) \in P_{L_{init_1}}$.
By Def. 10 we have $i_{1P}(P_I) \subseteq OUT(K_1)$ and by Def. 7 we have that $in_{1P} \circ proj(K_1)_P : P_{L_{init_1}} \to Skel(K_1)$ is bijective, which implies that for $(a, i_{1P}(p)) \in P_{L_{init_1}}$ follows that $(a, i_{1P}(p)) \in OUT(L_{init_1})$. Per construction of J follows that $(a,p) \in P_J$ and hence $k_2(y) \in P_J$.
For $k = k_2$ we have:
$in_{IP} \circ k(y) = k(y) = k_2(y)$ (5) commutes.
and
$in_{1P} \circ k_1 = Flat(i_1)_P \circ k_2 = Flat(i_1)_P \circ i_{IP} \circ k = in_{1P} \circ j_{1P} \circ k$
and since in_{1P} is monomorphism, this implies
$k_1 = j_{1P} \circ k$ (4) commutes

(c) Uniqueness:
Let us assume there exists $k' : Y \to P_J$ with $j_{1P} \circ k' = k_1$ and $in_{IP} \circ k' = k_2$. Then follows
$in_{IP} \circ k' = k_2 = in_{IP} \circ k$
and due to the fact that in_{IP} is injective we have $k' = k$. Hence k is unique.

Since the sets of places and transitions are componentwise Pullbacks in **SET**, (1) is Pullback in **PTNet**.

Proof (Theorem 2 in Section 4).
We have to show that

1. K is an AHL-occurrence net and
2. all $L_{init} \in INS$ are instantiations of K:
   (a) $L_{init} \subseteq Flat(K)$, i.e. there exists an injection $in : L_{init} \to Flat(K)$ and
   (b) the projection $proj(K) \circ in : L_{init} \to Skeleton(K)$ is an isomorphism.

Proof (Part 1). Follows directly from Theorem 1 in Section 4.

Proof (Part 2). Given the Pushout (1) we obtain the instantiation interface J as the Pullback (2) of $L_{init_1} \xrightarrow{in_1} Flat(K_1) \xleftarrow{Flat(i_1)} Flat(I)$ in **PTNet** using Lemma 1.

$$\begin{tikzcd} I \arrow[r, "i_1"] \arrow[d, "i_2"'] \arrow[dr, phantom, "(1)"] & K_1 \arrow[d, "i_1'"] \\ K_2 \arrow[r, "i_2'"'] & K \end{tikzcd}$$

$$\begin{tikzcd} & L_{init_1} \arrow[rr, "in_1"] \arrow[dr, phantom, "(2)"] & & Flat(K_1) \\ J \arrow[ur, "j_1"] \arrow[rr, "in_I"'] & & Flat(I) \arrow[ur, "Flat(i_1)"'] & \end{tikzcd}$$

Since Pullbacks are closed under monomorphisms, the injectivity of in_1 and $Flat(i_1)$ lead to the injectivity of $j_1 : J \to L_{init}$ and $in_I : J \to Flat(I)$.
First we will show in Lemma 2 that the projection $proj(I) \circ in_I$ is an isomorphism.

Lemma 2. *The projection $proj(I) \circ in_I : J \to Skel(I)$ is an isomorphism.*

Proof (Lemma 2). Injectivity of $proj(I) \circ in_I$:
Since $proj$ is a natural transformation, diagramm (3) commutes

$$\begin{tikzcd}
& L_{init_1} \arrow[rr, "in_1"] & & Flat(K_1) \arrow[rr, "proj(K_1)"] & & Skel(K_1) \\
& & \textbf{(2)} & & \textbf{(3)} & \\
J \arrow[uur, "j_1"] \arrow[rr, "in_I"'] & & Flat(I) \arrow[uur, "Flat(i_1)"'] \arrow[rr, "proj(I)"'] & & Skel(I) \arrow[uur, "Skel(i_1)"'] &
\end{tikzcd}$$

and hence $proj(K_1) \circ in_1 \circ j_1 = Skel(i_1) \circ proj(I) \circ in_I$.
$proj(K_1) \circ in_1 \circ j_1$ is injective, because j_1 is injective and $proj(K_1) \circ in_1$ is isomorphic, and hence $Skel(i_1) \circ proj(I) \circ in_I$ is injective. This implies that $proj(I) \circ in_I$ is injective.

Surjectivity of $proj(I) \circ in_I$:
Since I has no transitions, it suffices to show that $(proj(I) \circ in_I)_P$ is surjective, i.e. for all $x \in P_{Skel(I)}$ there exists $y \in P_J$, s.t. $(proj(I) \circ in_I)_P(y) = x$.
Let $x \in P_{Skel(I)}$. Then we have $i_{1P}(x) \in P_{Skel(K_1)}$. Since $(proj(K_1) \circ in_1)_P$ is bijective because L_{init_1} is an instantiation, there exists an unique $a \in A_{type(x)}$, s.t. $(a, i_{1P}(x)) \in P_{L_{init_1}}$. Furthermore $x \in P_{Skel(I)}$ implies $x \in P_I$ and hence by Def. 17 there is also $(a, x) \in P_{Flat(I)}$ with $proj(I)_P(a, x) = x$ and $Flat(i_1)_P(a, x) = (a, i_1(x)) \in P_{Flat(K_1)}$.
The fact that $(a, i_{1P}(x)) \in P_{L_{init_1}}$ with $in_1(a, i_{1P}(x)) = (a, i_{1P}(x)) \in P_{Flat(K_1)}$ implies because of the Pullback-property that there exists $y \in P_J$, s.t. $in_I(y) = (a, x) \in P_{Flat(I)}$.
Hence we have $(proj(I) \circ in_I)_P(y) = proj(I)_P(in_{IP}(y)) = proj(I)_P(a, x) = x$.

$$\begin{tikzcd}
& (a, i_{1,P}(x)) \arrow[rr, mapsto, "in_1"] & & (a, i_{1,P}(x)) \arrow[rr, mapsto, "proj(K_1)"] & & i_{1,P}(x) \\
& & \textbf{(2)} & & \textbf{(3)} & \\
y \arrow[uur, mapsto, "j_1"] \arrow[rr, "in_I"'] & & (a, x) \arrow[uur, mapsto, "Flat(i_1)"'] \arrow[rr, mapsto, "proj(I)"'] & & x \arrow[uur, mapsto, "Skel(i_1)"'] &
\end{tikzcd}$$

Since $proj(I) \circ in_I$ is injective and surjective, it also is bijective.

$in \circ proj(K)$ is isomorphic:
We construct L_{init} as Pushout of j_1 and j_2 leading to an unique $in : L_{init} \to Flat(K)$, s.t. the following cube commutes:

$$\begin{tikzcd}
 & L_{init1} \arrow[rr, "in_1"] \arrow[ddr] & \text{(PB)} & Flat(K_1) \arrow[ddr] & \\
J \arrow[ur, "j_1"] \arrow[rr, "in_I"] \arrow[ddr, "j_2"'] & & Flat(I) \arrow[ur, "Flat(i_1)"] \arrow[ddr, "Flat(i_2)"'] & & \text{(PO2)} \\
 & \text{(PO1)} & L_{init} \arrow[rr, "in"] & & Flat(K) \\
 & L_{init2} \arrow[ur] \arrow[rr, "in_2"] & & Flat(K_2) \arrow[ur] &
\end{tikzcd}$$

Also the following cube commutes because *proj* is a natural transformation:

$$\begin{tikzcd}
 & Flat(K_1) \arrow[rr, "proj(K_1)"] \arrow[ddr] & & Skel(K_1) \arrow[ddr] & \\
Flat(I) \arrow[ur, "Flat(i_1)"] \arrow[rr, "proj(K_1)"] \arrow[ddr, "Flat(i_2)"'] & & Skel(I) \arrow[ur, "Skel(i_1)"] \arrow[ddr, "Skel(i_2)"'] & & \text{(PO3)} \\
 & \text{(PO2)} & Flat(K) \arrow[rr, "proj(K)"] & & Skel(K) \\
 & Flat(K_2) \arrow[ur] \arrow[rr, "proj(K_2)"] & & Skel(K_2) \arrow[ur] &
\end{tikzcd}$$

Since $proj(K_1) \circ in_1$, $proj(K_2) \circ in_2$ and $proj(I) \circ in_I$ are isomorphisms and the following cube commutes, the fact that $Skel(K)$ is Pushout implies that $proj(K) \circ in$ is an isomorphism.

$$\begin{tikzcd}
 & L_{init1} \arrow[rr, "in_1"] \arrow[ddr] & \text{(PB)} & Flat(K_1) \arrow[rr, "proj(K_1)"] \arrow[ddr] & & Skel(K_1) \arrow[ddr] & \\
J \arrow[ur, "j_1"] \arrow[rr, "in_I"] \arrow[ddr, "j_2"'] & & Flat(I) \arrow[ur, "Flat(i_1)"] \arrow[rr, "proj(K_1)"] \arrow[ddr, "Flat(i_2)"'] & & Skel(I) \arrow[ur] \arrow[ddr] & & \text{(PO3)} \\
 & \text{(PO1)} & L_{init} \arrow[rr, "in"] & \text{(PO2)} & Flat(K) \arrow[rr, "proj(K)"] & & Skel(K) \\
 & L_{init2} \arrow[ur] \arrow[rr, "in_2"] & & Flat(K_2) \arrow[ur] \arrow[rr, "proj(K_2)"] & & Skel(K_2) \arrow[ur] &
\end{tikzcd}$$

Injectivity of *in*:
Due to the fact that $proj(K) \circ in$ is isomorphic, it is injective and hence *in* injective.

Remark 2. We can obtain an inclusion by renaming the elements of the instantiation by taking the image $\hat{L}_{init} = in(L_{init})$ which is isomorphic to L_{init} providing an inclusion $\hat{in} : \hat{L}_{init} \to Flat(K)$.

A.5 Proof of Theorem 3 in Section 4

Proof. By Theorem 2 in Section 4 the composition $(K, INIT, INS)$ of the AHL-occurrence nets with instantiations is an AHL-occurrence net with instantiations and hence $(K, INIT, INS)$ together with $mp : K \to N$ is an AHL-Process with instantiations, where mp is the unique morphism induced by Pushout (1) with

$$mp = (mp_P, mp_T)$$

$$mp_P(p) = \begin{cases} mp_{1P}(p_1) & p = i'_{1P}(p_1) \\ mp_{2P}(p_2) & p = i'_{2P}(p_2) \end{cases}$$

$$mp_T(t) = \begin{cases} mp_{1T}(t_1) & t = i'_{1T}(t_1) \\ mp_{2T}(t_2) & t = i'_{2T}(t_2) \end{cases}$$

A.6 Proof of Theorem 4 in Section 4

Proof. We have to show that

1. there are bijections $e_P : P_K \to P_{K'}$ and $e_T : T_K \to T_{K'}$, s.t. the diagram in Def. 12 commutes componentwise and
2. the instantiations INS and INS' are equivalent using that the instantiations INS_1 and INS_2 are consistent.

Proof of Part 1. Let us define gluing points $GP(P_K)$ and $GP(P_{K'}$ by

- $GP(P_K) = (i'_1 \circ i_3(P_I) \cup i'_2 \circ i_4(P_I) \cup i'_1 \circ i_1(P_I))$ where $i'_1 \circ i_1(P_I) = i'_2 \circ i_2(P_I)$
- $GP(P_{K'}) = (i'_4 \circ i_2(P_I) \cup i'_3 \circ i_1(P_I) \cup i'_3 \circ i_3(P_I))$ where $i'_3 \circ i_3(P_I) = i'_4 \circ i_4(P_I)$.

First we show that the three components of $GP(P_K)$ are disjoint which follows symmetrically for $GP(P_{K'})$.

$$i'_1 \circ i_3(P_I) \cap i'_1 \circ i_1(P_I) = i'_1(i_3(P_I) \cap i_1(P_I)) \subseteq i'_1(IN(K_1) \cap OUT(K_1)) = \emptyset$$

by assumption $IN(K_1) \cap OUT(K_1) = \emptyset$.

$$i'_2 \circ i_4(P_I) \cap i'_2 \circ i_2(P_I) = i'_2(i_4(P_I) \cap i_2(P_I)) \subseteq i'_2(OUT(K_2) \cap IN(K_2)) = \emptyset$$

by assumption $IN(K_2) \cap OUT(K_2) = \emptyset$.

For the third intersection we need

Lemma 3. *$IN(K) \cap OUT(K) = \emptyset$ and similar $IN(K') \cap OUT(K') = \emptyset$.*

Using Lemma 3 we can show $i'_1 \circ i_3(P_I) \cap i'_2 \circ i_4(P_I) = \emptyset$:

$$i'_1 \circ i_3(P_I) \cap i'_2 \circ i_4(P_I) \subseteq i'_1(IN(K_1)) \cap i'_2(OUT(K_2)) \subseteq IN(K) \cap OUT(K) = \emptyset$$

Proof of Lemma 3 Assume that there exists $p \in IN(K) \cap OUT(K)$, we have

$$IN(K) \subseteq i'_1(IN(K_1)) \cup i'_2(IN(K_2))$$

and

$$OUT(K) \subseteq i'_1(OUT(K_1)) \cup i'_2(OUT(K_2)).$$

Case 1 $p \in i'_1(IN(K_1)) \cap i'_1(OUT(K_1)) = i'_1(IN(K_1)) \cap OUT(K_1)) = \emptyset$ (contradiction)

Case 2 $p \in i'_2(IN(K_2)) \cap i'_2(OUT(K_2)) = \emptyset$ (contradiction)
Case 3

$$\begin{aligned} & p \in i'_1(IN(K_1)) \cap i'_2(OUT(K_2)) \\ \Rightarrow & \exists p_1 \in IN(K_1), p_2 \in OUT(K_2) \text{ with } p = i'_1(p_1) = i'_2(p_2) \\ \Rightarrow & \exists p \in P_I : i_1(p) = p_1, i_2(p) = p_2 \text{ by pushout (1)} \\ \Rightarrow & i_1(p) = p_1 \in OUT(K_1) \end{aligned}$$

contradicts $p_1 \in IN(K_1)$ and $IN(K_1) \cap OUT(K_1) = \emptyset$.
Case 4 symmetric to Case 3.

This implies that $GP(P_K)$ is the disjoint union

$$GP(P_K) = GP_1(P_K) \cup GP_2(P_K) \cup GP_3(P_K)$$

with $GP_x(P_K)$ for $x = 1, 2, 3$ defined below.

Now we are able to define $e_P : P_K \to P_{K'}$ via the non gluing points NGP. Let $NGP(P_K) = P_K \setminus GP(P_K)$ then we have the following disjoint unions:

$$\begin{aligned} P_K &= GP_1(P_K) \cup GP_2(P_K) \cup GP_3(P_K) \cup NGP(P_K) \\ P_{K'} &= GP_1(P_{K'}) \cup GP_2(P_{K'}) \cup GP_3(P_{K'}) \cup NGP(P_{K'}) \end{aligned}$$

with

$$\begin{aligned} GP_1(P_K) = i'_1 \circ i_3(P_I), GP_1(P_{K'}) = i'_4 \circ i_2(P_I) \\ GP_2(P_K) = i'_2 \circ i_4(P_I), GP_2(P_{K'}) = i'_3 \circ i_1(P_I) \\ GP_3(P_K) = i'_1 \circ i_1(P_I), GP_3(P_{K'}) = i'_3 \circ i_3(P_I) \end{aligned}$$

and we define e_P by $e_{P_x} : GP_x(P_K) \to GP_x(P_{K'})$ for $x = 1, 2, 3$ for all $p \in P_I$ by

5. $e_{P_1}(i'_1 \circ i_3(p)) = i'_4 \circ i_2(p)$
6. $e_{P_2}(i'_2 \circ i_4(p)) = i'_3 \circ i_1(p)$
7. $e_{P_3}(i'_1 \circ i_1(p)) = i'_3 \circ i_3(p)$

which are bijections because all morphisms are injective. Moreover we have $GP(P_{K_1}) = i_1(P_I) \cup i_3(P_I)$ and $GP(P_{K_2}) = i_2(P_I) \cup i_4(P_I)$.

Now let $NGP(P_{K_x}) = P_{K_x} \setminus GP(P_{K_x})$ for $x = 1, 2$ then we have by Lemma 4 below:

$$\begin{aligned} NGP(P_K) = i'_1(NGP(P_{K_1})) \cup i'_2(NGP(P_{K_2})) = NGP_1(P_K) \cup NGP_2(P_K) \\ NGP(P_{K'}) = i'_3(NGP(P_{K_1})) \cup i'_4(NGP(P_{K_2})) = NGP_1(P_{K'}) \cup NGP_2(P_{K'}) \end{aligned}$$

and we define for $p_1 \in NGP(P_{K_1})$ and $p_2 \in NGP(P_{K_2})$

8. $e_{P_4} : NGP_1(P_K) \to NGP_1(P_{K'})$ by $e_{P_4}(i'_1(p_1)) = i'_3(p_1)$
9. $e_{P_5} : NGP_2(P_K) \to NGP_2(P_{K'})$ by $e_{P_5}(i'_2(p_2)) = i'_4(p_2)$

which are bijections because all morphisms are injective.

Alltogether we have a bijection

$$e_P = e_{P_1} + e_{P_2} + e_{P_3} + e_{P_4} + e_{P_5} : P_K \to P_{K'}$$

Moreover there is a bijection $e_T : T_K \to T_{K'}$ because we have the following disjoint unions

$$T_K \cong T_{K_1} \uplus T_{K_2} \text{ and } T_{K'} \cong T_{K_1} \uplus T_{K_2}$$

Lemma 4. *$NGP(P_K) = P_K \setminus GP(P_K) = i'_1(NGP(P_{K_1})) \cup i'_2(NGP(P_{K_2}))$ and similar for $NGP(P_{K'})$.*

Proof of Lemma 4

$$\begin{aligned}
&i'_1(NGP(P_{K_1})) \cup i'_2(NGP(P_{K_2}))\\
&= i'_1(P_{K_1} \setminus GP(P_{K_1})) \cup i'_2(P_{K_2} \setminus GP(P_{K_2}))\\
&= i'_1(P_{K_1} \setminus (i_1(P_I) \cup i_3(P_I))) \cup i'_2(P_{K_2} \setminus (i_2(P_I) \cup i_4(P_I)))\\
&= i'_1(P_{K_1}) \setminus (i'_1 \circ i_1(P_I) \cup i'_1 \circ i_3(P_I)) \cup i'_2(P_{K_2}) \setminus (i'_2 \circ i_2(P_I) \cup i'_2 \circ i_4(P_I))\\
&= i'_1(P_{K_1}) \cup i'_2(P_{K_2}) \setminus (GP_1(P_K) \cup GP_2(P_K) \cup GP_3(P_K))\\
&= P_K \setminus GP(P_K)
\end{aligned}$$

It remains to show for Proof of 1. the following:

Lemma 5 (Compatibility of e_p and e_T with mp and mp'). *The following diagram commutes componentwise.*

$$\begin{tikzcd}
K \arrow[dd, shift right, "e_P"'] \arrow[dd, shift left, "e_T"] \arrow[dr, "mp"] & \\
& AN \\
K' \arrow[ur, "mp'"'] &
\end{tikzcd}$$

Proof of Lemma 5 mp and mp' are defined by the induced morphisms of pushout (1) and (2), where the outer diagrams commute by compatibility of mp_1 and mp_2 with i_1, i_2, i_3 and i_4.

$$\begin{tikzcd}
& K_1 \arrow[dr, "i'_1"] \arrow[drr, "mp_1"] & & \\
I \arrow[ur, "i_1"] \arrow[dr, "i_2"'] & \textbf{(1)} & K \arrow[r, "mp"] & AN \\
& K_2 \arrow[ur, "i'_2"] \arrow[urr, "mp_2"'] & &
\end{tikzcd}$$

$$\begin{tikzcd}
& K_2 \arrow[dr, "i'_4"] \arrow[drr, "mp_2"] & & \\
I \arrow[ur, "i_4"] \arrow[dr, "i_3"'] & \textbf{(2)} & K' \arrow[r, "mp'"] & AN \\
& K_1 \arrow[ur, "i'_3"] \arrow[urr, "mp_1"'] & &
\end{tikzcd}$$

1. The bijection $e_T : T_K \to T_{K'}$ is induced by id_1 and id_2 in the following diagrams (3) and (4), s.t. (5) commutes if the outer diagrams commute.

$$\begin{tikzcd}
T_{K_1} \arrow[rr, "i'_1"] \arrow[dd, "id_1"'] & & T_K \arrow[drr, "mp"] \arrow[dd, "e_T"] & & \\
& (\mathbf{3}) & & (\mathbf{5}) & T_{AN} \\
T_{K_1} \arrow[rr, "i'_3"'] & & T_{K'} \arrow[urr, "mp'"'] & &
\end{tikzcd}
\qquad
\begin{tikzcd}
T_{K_2} \arrow[rr, "i'_2"] \arrow[dd, "id_2"'] & & T_K \arrow[drr, "mp"] \arrow[dd, "e_T"] & & \\
& (\mathbf{4}) & & (\mathbf{5}) & T_{AN} \\
T_{K_2} \arrow[rr, "i'_4"'] & & T_{K'} \arrow[urr, "mp'"'] & &
\end{tikzcd}$$

The outer diagrams commute because we have $mp \circ i'_1 = mp_1 = mp' \circ i'_3$ and $mp \circ i'_2 = mp_2 = mp' \circ i'_4$.

2. The bijection $e_P : P_K \to P_{K'}$ is given by $e_{P_1} + e_{P_2} + e_{P_3} + e_{P_4} + e_{P_5}$. Note that the bijections e_{P_x} are defined by commutativity of diagrams (x) for $x = 1, \ldots, 5$ and the required digram (6) commutes if all the outer diagram commute.

(a)

$$\begin{aligned}
mp \circ i'_1 \circ i_3 &= mp_1 \circ i_3 \\
&= mp_2 \circ i_2 \qquad \text{(by compatibility)} \\
&= mp' \circ i'_4 \circ i_2
\end{aligned}$$

$$\begin{tikzcd}
& & GP_1(P_K) \arrow[r, hook] \arrow[dd, "e_{P_1}"] & P_K \arrow[drr, "mp"] \arrow[dd, "e_P"'] & & \\
P_I \arrow[urr, "i'_1 \circ i_3"] \arrow[drr, "i'_4 \circ i_2"'] & (\mathbf{1}) & & & (\mathbf{6}) & P_{AN} \\
& & GP_1(P_{K'}) \arrow[r, hook] & P_{K'} \arrow[urr, "mp'"'] & &
\end{tikzcd}$$

(b)

$$\begin{aligned}
mp \circ i'_2 \circ i_4 &= mp_2 \circ i_4 \\
&= mp_1 \circ i_1 \qquad \text{(by compatibility)} \\
&= mp' \circ i'_3 \circ i_1
\end{aligned}$$

$$\begin{tikzcd}
& & GP_2(P_K) \arrow[r, hook] \arrow[dd, "e_{P_2}"] & P_K \arrow[drr, "mp"] \arrow[dd, "e_P"'] & & \\
P_I \arrow[urr, "i'_2 \circ i_4"] \arrow[drr, "i'_3 \circ i_1"'] & (\mathbf{2}) & & & (\mathbf{6}) & P_{AN} \\
& & GP_2(P_{K'}) \arrow[r, hook] & P_{K'} \arrow[urr, "mp'"'] & &
\end{tikzcd}$$

(c)

$$\begin{aligned}
mp \circ i'_1 \circ i_1 &= mp_1 \circ i_1 \\
&= mp_2 \circ i_4 \qquad \text{(by compatibility)} \\
&= mp' \circ i'_4 \circ i_4 \\
&= mp' \circ i'_3 \circ i_3
\end{aligned}$$

$$\begin{tikzcd}
 & GP_3(P_K) \arrow[r, hook] \arrow[dd, "e_{P_3}"] & P_K \arrow[dd, "e_P"'] \arrow[dr, "mp"] & \\
P_I \arrow[ur, "i_1' \circ i_1"] \arrow[dr, "i_3' \circ i_3"'] \arrow[r, phantom, "(\mathbf{3})"] & & & P_{AN} \arrow[l, phantom, "(\mathbf{6})"] \\
 & GP_3(P_{K'}) \arrow[r, hook] & P_{K'} \arrow[ur, "mp'"'] &
\end{tikzcd}$$

(d)

$$\begin{aligned} mp \circ i_1' &= mp_1 \\ &= mp' \circ i_3' \end{aligned}$$

$$\begin{tikzcd}
 & NGP_1(P_K) \arrow[r, hook] \arrow[dd, "e_{P_4}"] & P_K \arrow[dd, "e_P"'] \arrow[dr, "mp"] & \\
NGP(P_{K_1}) \arrow[ur, "i_1'"] \arrow[dr, "i_3'"'] \arrow[r, phantom, "(\mathbf{4})"] & & & P_{AN} \arrow[l, phantom, "(\mathbf{6})"] \\
 & NGP_1(P_{K'}) \arrow[r, hook] & P_{K'} \arrow[ur, "mp'"'] &
\end{tikzcd}$$

(e)

$$\begin{aligned} mp \circ i_2' &= mp_2 \\ &= mp' \circ i_4' \end{aligned}$$

$$\begin{tikzcd}
 & NGP_2(P_K) \arrow[r, hook] \arrow[dd, "e_{P_5}"] & P_K \arrow[dd, "e_P"'] \arrow[dr, "mp"] & \\
NGP(P_{K_2}) \arrow[ur, "i_2'"] \arrow[dr, "i_4'"'] \arrow[r, phantom, "(\mathbf{5})"] & & & P_{AN} \arrow[l, phantom, "(\mathbf{6})"] \\
 & NGP_2(P_{K'}) \arrow[r, hook] & P_{K'} \arrow[ur, "mp'"'] &
\end{tikzcd}$$

Remark 3. For the existence of mp and mp' we need already the compatibilities $mp_1 \circ i_1 = mp_2 \circ i_2$ and $mp_1 \circ i_3 = mp_2 \circ i_4$. In 2(a) - 2(c) we need in addition that all these morphisms are equal.

Proof of Part 2. Given $L_{init} = L_{init_1} \circ_{(J,j_1,j_2)} L_{init_2}$ with pushout (3) we have by consistency of INS_1 and INS_2 $L_{init'} = L_{init_2'} \circ_{(J,j_4,j_3)} L_{init_1'}$ with pushout (4) and vice versa, s.t. properties 1-4 in Def. 14 are satisfied.

First we have to show according to Def. 13.2 that

$$\forall (a,p) \in A_{type(p)} \otimes P_K : (a,p) \in IN(L_{init}) \Leftrightarrow (a, e_P(p)) \in IN(L_{init'})$$

"$\Rightarrow$" Let $(a,p) \in IN(L_{init})$. By construction we have

$$IN(L_{init}) = j_1'(IN(L_{init_1})) \cup j_2'(IN(L_{init_2}) \setminus j_2(J))$$

and

$$IN(L_{init'}) = .j_4'(IN(L_{init_2'})) \cup j_3'(IN(L_{init_1'}) \setminus j_3(J)).$$

Case 1 Let $(a,p) \in j_1'(IN(L_{init_1}))$. Then there exists $(a,p_1) \in IN(L_{init_1})$ with

$$j_1'(a,p_1) = (a, i_1'(p_1)) = (a,p)$$

So we have $i_1'(p_1) = p$.

Case 1.1 Let $(a,p_1) \notin GP(L_{init_1})$. Then we have

$$\begin{aligned}
& (a,p_1) \in IN(L_{init_1}) \setminus GP(L_{init_1}) \subseteq IN(L_{init_1'}) \\
& \hspace{14em} \text{(by consistency)} \\
\Rightarrow\ & j_3'(a,p_1) = (a, i_3'(p_1)) \in IN(L_{init'}) \\
& \hspace{8em} \text{(because } p_1 \notin i_3(P_I) \text{ and Def. } IN(L_{init'})\text{)} \\
\Rightarrow\ & (a, e_P(p)) = (a, e_p \circ i_1'(p_1)) = (a, i_3'(p_1)) \in IN(L_{init'}) \\
& \hspace{8em} \text{(by } e_P \text{ commutes on } NGP \text{ (see 8.))} \\
\Rightarrow\ & (a, e_P(p)) \in IN(L_{init'})
\end{aligned}$$

Case 1.2 Let $(a,p_1) \in GP(L_{init_1})$.

Case 1.2.1 Let $p_1 = i_3(p_I)$ for $p_I \in P_I$.

$$\begin{aligned}
& (a,p_1) \in IN(L_{init_1}) \\
\Rightarrow\ & (a, i_3(p_I)) \in IN(L_{init_1}) \\
\Rightarrow\ & (a, i_2(p_I)) \in IN(L_{init_2'}) && \text{(by consistency (see 3.))} \\
\Rightarrow\ & j_4'(a, i_2(p_I)) \in IN(L_{init'}) && \text{(by Def.} IN(L_{init'})\text{)} \\
\Rightarrow\ & (a, i_4' \circ i_2(p_I)) \in IN(L_{init'})
\end{aligned}$$

Then we have $(a, e_P(p)) = (a, e_p(i_1' \circ i_3(p_I)) = (a, i_4' \circ i_2(p_I)) \in IN(L_{init'})$ using $p = i_1' \circ i_3(p_I) \Rightarrow e_p(p) = i_4' \circ i_2(p_I)$ (see 5.).

Case 1.2.2 Let $p_1 = i_1(p_I)$ for $p_I \in P_I$. Then $p_1 \in OUT(K_1) \cap IN(K_1)$ which is a contradiction to the assumption that

$$OUT(K_1) \cap IN(K_1) = \emptyset$$

Case 2 Let $(a,p) \in j_2'(IN(L_{init_2}) \setminus j_2(J))$. Then there exists $(a,p_2) \in IN(L_{init_2}) \setminus j_2(J)$ with

$$j_2'(a,p_2) = (a, i_2'(p_2)) = (a,p)$$

So we have $i_2'(p_2) = p$.

Case 2.1 Let $(a,p_2) \notin GP(L_{init_2})$. Then we have

$$\begin{aligned}
& (a,p_2) \in IN(L_{init_2}) \setminus GP(L_{init_2}) \subseteq IN(L_{init_2'}) \\
& \hspace{10em} \text{(by consistency (see 1.))} \\
\Rightarrow\ & j_4'(a,p_2) = (a, i_4'(p_2)) \in IN(L_{init'}) \\
& \hspace{10em} \text{(by Def. } IN(L_{init'})\text{)} \\
\Rightarrow\ & (a, e_P(p)) = (a, e_p \circ i_2'(p_2)) = (a, i_4'(p_2)) \in IN(L_{init'}) \\
& \hspace{8em} \text{(by } e_P \text{ commutes on } NGP \text{ (see 9.))} \\
\Rightarrow\ & (a, e_P(p)) \in IN(L_{init'})
\end{aligned}$$

Case 2.2 Let $(a,p_2) \in GP(L_{init_2})$.

Case 2.2.1 Let $p_2 \in i_2(I)$. Then $(a, p_2) \in j_2(J)$ which is a contradiction to the assumption $(a, p) \in j'_2(j_2(J))$ which implies $(a, p_2) \notin j_2(J)$.

Case 2.2.2 Let $p_2 \in i_4(I)$.Then $p_2 \in OUT(K_2) \cap IN(K_2)$ which is a contradiction to the assumption that $OUT(K_2) \cap IN(K_2) = \emptyset$.

"⇐" Let $(a, p) \in IN(L_{init'})$ for $p \in P_{K'}$. We have to show $(a, e_p^{-1}(p)) \in IN(L_{init})$. By construction we have

$$IN(L_{init'}) = j'_4(IN(L_{init'_2})) \cup j'_3(IN(L_{init'_1}) \setminus j_3(J)).$$

Case 1 Let $(a, p) \in j'_4(IN(L_{init'_2}))$. Then there exists $(a, p_2) \in IN(L_{init'_2})$ with

$$j'_4(a, p_2) = (a, i'_4(p_2)) = (a, p)$$

So we have $i'_4(p_2) = p$.

Case 1.1 Let $(a, p_2) \notin GP(L_{init'_2})$. Then we have

$$\begin{aligned}
& (a, p_2) \in IN(L_{init'_2}) \setminus GP(L_{init'_2}) \subseteq IN(L_{init_2}) \\
& \qquad\qquad\qquad\qquad\qquad\qquad \text{(by consistency)} \\
\Rightarrow\ & j'_2(a, p_2) = (a, i'_2(p_2)) \in IN(L_{init}) \\
& \qquad\qquad\qquad \text{(because } p_2 \notin i_3(P_I) \text{ and Def. } IN(L_{init})\text{)} \\
\Rightarrow\ & (a, e_P^{-1}(p)) = (a, e_P^{-1} \circ i'_4(p_2)) = (a, i'_2(p_2)) \in IN(L_{init}) \\
& \qquad\qquad\qquad \text{(by } e_P \text{ commutes on } NGP \text{ (see 9.))} \\
\Rightarrow\ & (a, e_P(p)) \in IN(L_{init})
\end{aligned}$$

Case 1.2 Let $(a, p_2) \in GP(L_{init'_2})$.

Case 1.2.1 Let $p_2 = i_2(p_I)$ for $p_I \in P_I$.

$$\begin{aligned}
& (a, p_2) \in IN(L_{init'_2}) \\
\Rightarrow\ & (a, i_2(p_I)) \in IN(L_{init'_2}) \\
\Rightarrow\ & (a, i_3(p_I)) \in IN(L_{init_1}) && \text{(by consistency)} \\
\Rightarrow\ & j'_1(a, i_3(p_I)) \in IN(L_{init}) && \text{(by Def.} IN(L_{init}) \\
\Rightarrow\ & (a, i'_1 \circ i_3(p_I)) \in IN(L_{init})
\end{aligned}$$

Then we have

$$(a, e_P^{-1}(p)) = (a, e_p^{-1}(i'_4 \circ i_2(p_I)) = (a, i'_1 \circ i_3(p_I)) \in IN(L_{init})$$

using $p = i'_1 \circ i_3(p_I) \Rightarrow e_p(p) = i'_4 \circ i_2(p_I)$ (see 5.).

Case 1.2.2 Let $p_2 = i_4(p_I)$ for $p_I \in P_I$. Then $p_2 \in OUT(K_2) \cap IN(K_2)$ which is a contradiction to the assumption that

$$OUT(K_2) \cap IN(K_2) = \emptyset$$

Case 2 Let $(a, p) \in j'_3(IN(L_{init'_1}) \setminus j_3(J))$. Then there exists $(a, p_1) \in IN(L_{init'_1}) \setminus j_3(J)$ with

$$j'_3(a, p_1) = (a, i'_3(p_1)) = (a, p)$$

So we have $i'_3(p_1) = p$.

Case 2.1 Let $(a, p_1) \notin GP(L_{init'_1})$. Then we have

$$\begin{aligned}
& (a, p_1) \in IN(L_{init'_1}) \setminus GP(L_{init'_1}) \subseteq IN(L_{init_1}) \\
& \qquad\qquad\qquad\qquad \text{(by consistency)} \\
\Rightarrow\ & j'_1(a, p_1) = (a, i'_1(p_1)) \in IN(L_{init}) \\
& \qquad\qquad\qquad\qquad \text{(by Def. } IN(L_{init})\text{)} \\
\Rightarrow\ & (a, e_P^{-1}(p)) = (a, e_P^{-1} \circ i'_3(p_1)) = (a, i'_1(p_1)) \in IN(L_{init}) \\
& \qquad\qquad\qquad\qquad \text{(by } e_P \text{ commutes on } NGP \text{ (see 8.))} \\
\Rightarrow\ & (a, e_P^{-1}(p)) \in IN(L_{init})
\end{aligned}$$

Case 2.2 Let $(a, p_1) \in GP(L_{init'_1})$.

Case 2.2.1 Let $p_1 \in i_3(I)$. Then $(a, p_1) \in j_3(J)$ which is a contradiction to the assumption $(a, p) \notin j'_3(j_3(J))$ which implies $(a, p_1) \notin j_3(J)$.

Case 2.2.2 Let $p_1 \in i_1(I)$.Then $p_1 \in OUT(K_1) \cap IN(K_1)$ which is a contradiction to the assumption that $OUT(K_1) \cap IN(K_1) = \emptyset$.

Moreover we have to show according to Def. 13.2 that

$$\forall (a, p) \in A_{type(p)} \otimes P_K : (a, p) \in OUT(L_{init}) \Leftrightarrow (a, e_P(p)) \in OUT(L_{init'})$$

This can be shown analogously to **Case 1** and **Case 2** above.