Petri Net Transformations in the ≫Petri Net Baukasten≪*

B. Braatz, H. Ehrig, M. Urbášek

Technical University Berlin, Germany
Institute for Software Technology and Theoretical Computer Science
{bbraatz, ehrig, urbasek}@cs.tu-berlin.de

Abstract. The purpose of this contribution is to give an overview of constructions and results for Petri net transformations in the ≫Petri Net Baukasten≪ developed by the "DFG-Forschergruppe Petrinetz-Technologie". The two main concepts of Petri net transformations considered in this context are net class and net model transformations. In both cases we present first the relevance of transformations in the application developer view and them the technical constructions and results in the expert view of the ≫Petri Net Baukasten≪. Net class transformations are transformations between different Petri net classes, like elementary nets, place/transition nets and algebraic high-level nets. Net model transformations on the other hand are transformations of the net structure for nets within one Petri net class, like place or transition refinement. The main technical results are concerning the preservation of safety resp. liveness properties of net model transformations and compatibility results between net class and net model transformations. The relevance of the constructions and results for the application developer view is demonstrated by a small case study modeling the interaction of a buffer, a printer and a communication unit consisting of secure and non-secure channels. Finally we give an overview of other concepts of transformations: On one hand, transformations between Petri nets and other system modeling techniques, and on the other hand transformations between different net representation formats using XML-schemes and DTD-standards.

1 Introduction

The ≫Petri Net Baukasten≪ presented in [WER99,DFG99,GE01] and in chapter 1 of this volume, provides a unified presentation with different views on theory, applications and tools of Petri nets: the Expert View, the Application Developer

* This work is a part of the joint research project "DFG-Forschergruppe Petrinetz-Technologie" between H. Weber (Coordinator), H. Ehrig (both from the Technical University Berlin) and W. Reisig (Humboldt University Berlin), supported by the German Research Council (DFG).

View and the Tool Developer View. In this paper we show that different kinds of Petri net transformations play an important role in these three views.

In the Application Developer View net transformations are essential for realizing stepwise system development within the given process models. These process models present a methodology in the way Petri nets are to be used in the different steps during software development. They typically start with an abstract model of the system which is refined in further development steps. Refinement here means integration of sytem aspects like time, roles, data, etc. as well as modification of these aspects for incorporating exception handling, business rules, etc.

Transformations in the Tool Developer View mainly concern the date structures used in Petri net tools. They yield an exchange format in order to couple different tools. Thus, Petri net models created with one tool can be transferred to another tool. Consequently, algorithms of the target tool can be employed which have not been available in the source tool. Thus, transformations in this sense are used to link tools together, which enables the development of tool environments.

In the Expert View transformations are used to perform modifications of a net. They are formalized on a rigorous mathematical foundation, see [GPP01b]. For a systematic study of such Petri net transformations we distinguish two levels of transformations, called *net model transformations* and *net class transformations*. We consider model transformations given by rule-based modifications in the context of High-Level Replacement (HLR) Systems [EHKP91,EGP99] and class transformations by functors in the sense of category theory [AHS90]. The purpose of the formal transformations is twofold:

- On one hand net class transformations extend the theory of a given net class in the following sense: Petri net operations in the target net class are made available also for the source net class by transforming the net class, performing the operation, and subsequently interpreting back the result of the operation in the source net class.
- On the other hand both kinds of transformations together allow arbitrary modifications of a (start) net. Therefore, they are suitable to support stepwise enhancement of nets in the context of system development. In this sense, they yield a formal support of the Petri net based process models in the Application Developer View.

1.1 Overview of the Results

The following results concerning transformations have been achieved in the "DFG-Forschergruppe Petrinetz-Technologie", where an overview of the main constructions and results are presented in more detail in sections 2 and 3 of this paper.

1. **Classification of transformations w. r. t. preservation of system properties**
   Preservation of behavioral properties like liveness, deadlock freedom, or safety properties are important during the development of the system. In order to ensure preservation of these properties by transformations, both net class transformations and net model transformations have to be classified according to the preservation of these properties.
   In the case of net model transformations preservation of system properties leads to rule-based refinement introduced in [Pad96,Pad99]. Rule-based refinement has already been employed for the preservation of safety properties in [PGE98,PG98,PGH99,PHG00] and for liveness-preserving transformations in [GPU01]. Net class transformations have to be treated seperately. There are already results in the literature concerning preservation of properties by functors, e. g. [Lil95], which however do not cover all system properties and are only formulated for particular functors.
2. **Compatibility of structuring and transformation**
   For large systems adequate structuring techniques for composing subsystems and decomposing are indispensable. For transformation of subsystems and composition of transformed models, structuring is needed to be compatible with transformation.
   On the level of model transformations this requirement is already given, see e. g. [PER95,EGP99]. On the level of net class transformations compatibility is granted if the net class functor preserves colimits, see [GPP00].
3. **Consistency between models of different levels of abstraction**
   In Petri net based process models several models of the system are achieved on different levels of abstraction. Of course, all of these models should be consistent with each other, meaning that each model is a proper enhancement of the start model. This requirement can be achieved if model transformations are preserved by net class transformations between different levels of abstraction.
   The corresponding results have been proved in [GPP00,GPP01b].

1.2 Introduction to the Case Study

To illustrate the relevance of Petri net transformations in the application developer view of the »Petri Net Baukasten« we will model a simple printing system and show how the development of this model can be structured with net model and net class transformations.

The system to be modelled consists of a buffer, a printer and a communication unit. The printer can only print one task at the same time. The communication unit has two channels, one secure channel and one channel for which the integrity of the printing task has to be verified after the transmission by comparing two copies of the task, which are sent independently.

In Sect. 2 two ways of obtaining a place/transition net model of this system using stepwise net model transformations are explored. First, the system is set up by introducing the components buffer, printer and communication unit

separately and using gluing transformations to connect the components. Secondly, the system is introduced as a raw model and refinement rules are used to elaborate the printing and the transmission.

In Sect. 3 we show how the modelling of the system can be done using different Petri net classes on different layers of abstraction. First, a simple elementary net model is created and then transformed to place/transition nets, where the model of Sect. 2 is obtained. Then, the place/transition net is transformed to algebraic high level nets. This allows us to specify that the two tasks sent over the insecure channel can really be compared for equality.

1.3 Organization of this Paper

In section 2 we present net model transformations. Concerning the Application Developer View we show how to use transformations for stepwise development of communication based systems and discuss safety and liveness preserving transformations in our case study. The formal framework for net model transformations and main results concerning preservation of safety and liveness are contributions to the Expert View.

Net class transformations are presented in section 3. The case study is extended as a contribution to the Application Developer View. Net class transformations are formalized by functors between suitable net class categories in the Expert View.

A more detailed presentation of net class transformations can be found in the paper [PP02] in this volume.

The subsections of sections 2 and 3 concerning the Application Developer View can be read independently of those concerning the Expert View.

Finally in section 4 we give an overview of other transformations in the ≫Petri Net Baukasten≪, especially concerning the Tool Developer View, and a conclusion sketching further interesting lines of research concerning Petri net transformations.

2 Net Model Transformations

Net model transformations play an important role in the ≫Petri Net Baukasten≪. In the following subsections we discuss net model transformations from the Application Developer View and the Expert View.

2.1 Application Developer View

The main focus of an application developer is how net model transformations can be used to build up an application. The major applicability domain of net model transformations is stepwise development of communication based systems, which will be discussed next.

Stepwise Development of Communication Based-Systems

The main idea of stepwise development of systems is to offer to an application developer a number of transformation rules for different classes of Petri nets. These rules describe how to change a net model in order to obtain another more elaborated, refined and expressive net model.

Each transformation step can be depicted as a transformation according to an appropriate rule. The sequence of transformations then provides a transformation from the initial net G to the final net H:

$$G = G_0 \stackrel{p_1}{\Longrightarrow} G_1 \stackrel{p_2}{\Longrightarrow} \ldots \stackrel{p_n}{\Longrightarrow} G_n = H.$$

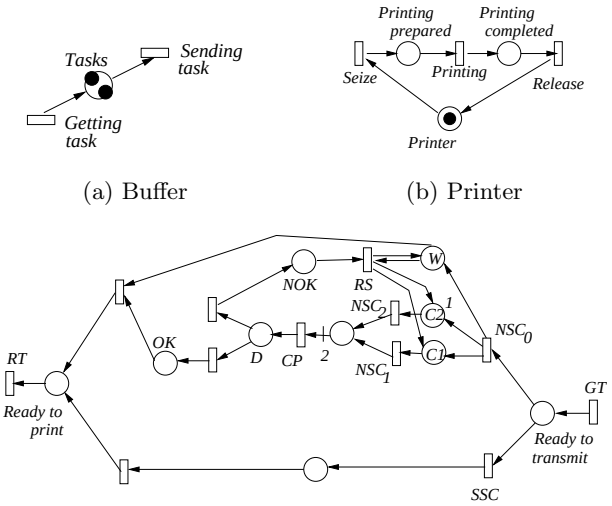
Each step $G_i \stackrel{p_{i+1}}{\Longrightarrow} G_{i+1}$ denotes a single transformation. The transformation process is rule-based and has a formal mathematical foundation, which is defined in the Expert View (see Section 2.2).

The preservation of suitable system properties during the transformation process is of interest in most applications, since the final model may be very large and hence difficult to check for certain system properties. It is of great importance for the application developer to check only the initial – usually quite small system – directly and to apply property preserving rules and transformations. For this reason structure and property preserving rules and transformations are supported in the Application Developer View of the ≫Petri Net Baukasten≪, especially horizontal structural techniques (union and fusion), safety-properties preserving rules for P/T, colored and AHL Petri nets, liveness preserving rules for P/T Petri nets.

A slightly different approach developer also in the "DFG- Forschergruppe Petrinetz-Technologie" is the transition refinement technique for distributed algorithms in [Peu01]. On one hand transition refinement in [Peu01], where a specific class of high-level nets is considered and system properties can be expressed by general temporal logic formulas, is more general than rule based transformation, where only a specific class of formulas is considered. On the other hand rule based transformations are more general concerning the applicability to a large variety of low- and high-level nets classes, which can be considered as instantiations of high-level replacement systems (see [Pad96]). Safety property preserving transformations have been developed for low- and high-level nets [GHP99,PG00b,PGE98], liveness preserving transformations, however up to now for P/T nets only [GPU01].

Case Study: Simple Communication Based System

We illustrate the rule-based stepwise development of a simple communication based system constructed as interconnection of three components: a buffer with two tasks, a printer and a communication unit (network) between buffer and printer consisting of a secure and non-secure channel. The components are depicted on Figure 1.



(a) Buffer (b) Printer

(c) Communication unit

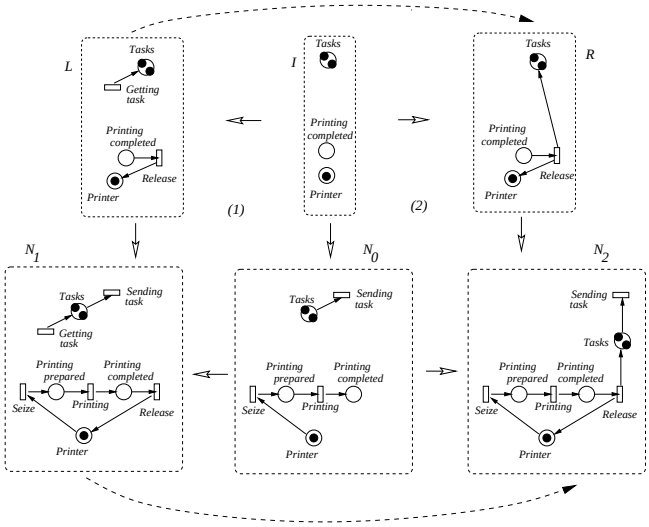
Fig. 1. Components of the system

The behavior of the printer and the buffer are obvious from the figure. The communication unit serves a secure or non-secure communication from the buffer to the printer.

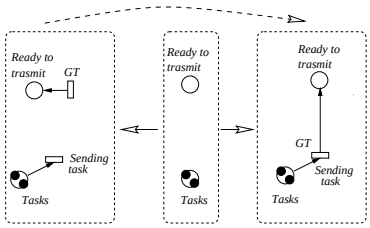
This unit receives a task which has to be sent over a transition (**GT**). It can send the message through a secure channel (**SSC**) or via a non-secure one (**NSC_0**, **NSC_1**, **NSC_2**). On the non-secure channel the message may become corrupted. Therefore, when a non-secure channel is used, two copies (**C1, C2**) of the message are to be sent and a transmission subunit waits (**W**) for an acknowledgment. A receiving subunit on the other end receives both copies and compares them (**CP**). (We assume that a message cannot be lost during the transfer.) If both copies are the same, the **OK** acknowledgement is sent back and the communication is ended. If both copies differ, then **NOK** acknowledgement is sent back and the transmission subunit resends (**RS**) the message again (in two copies). The two possible results of the comparison are modeled by a nondeterministic choice (conflict) in place **D**. The communication ends when a successful transfer is performed (**RT**).

The three components of the system shown in Figure 1 are interconnected by the application of the rules shown in Figure 2. In a first step we apply the buffer-printer rule in Figure 2a to the buffer and printer components in Figure 1. The corresponding transformation is shown in Figure 2a.

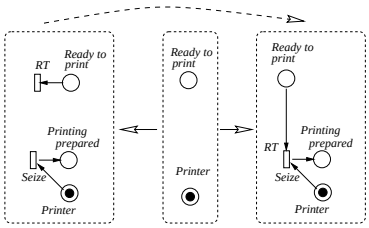
The rule buffer-printer is shown in the top row of Figure 2a. It consists of a left-hand side net L, a right-hand side net R and an interface net I. The transformation from net N_1 to net N_2 via this rule is constructed in two steps. In the first step we apply L to N_1 and remove all items of L in N_1 which do



(a) Buffer–Printer



(b) Buffer–Comm. unit



(c) Printer–Comm.unit

Fig. 2. Interconnection of components

not belong to the interface I. It leads to the intermediate net N_0. In the second step we glue together nets N_0 and R via the interface I leading to the net N_2. It is important to note that also N_1 can be considered as a result of a gluing construction, namely the gluing of N_0 and L along I such that the transformation in Figure 2a consists of two gluing constructions shown in diagrams (1) and (2) respectively according to the general construction of net model transformations in Section 2.2.

In a similar way we can apply the two other rules in Figures 2b,c to net N_2 combined with the communication unit net in Figure 1 leading to the net in Figure 3.

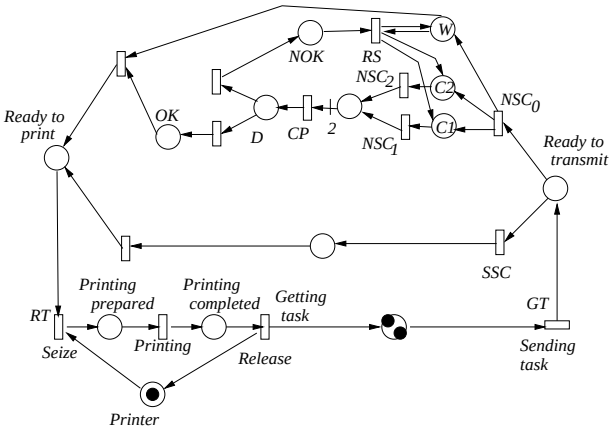


Fig. 3. Simple tasks' model

Finally we can apply the rule in Figure 4 modeling mutual exclusion between the secure and the non-secure channel leading to the final model in Figure 5.

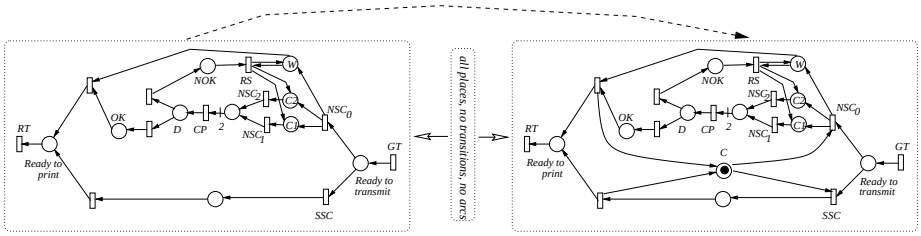


Fig. 4. Modeling exclusivity

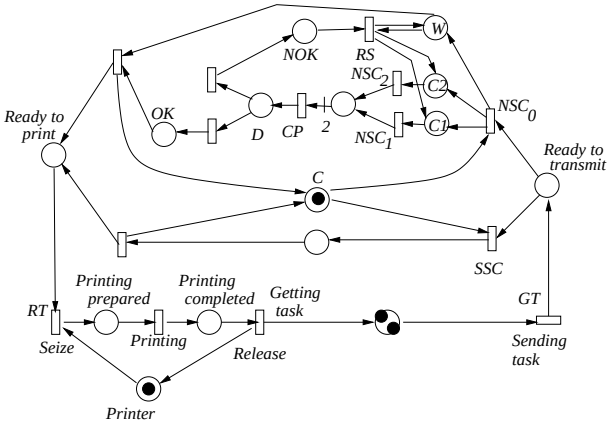


Fig. 5. Final model

Transformations Preserving Safety Properties

Now we want to show suitable safety properties for the final model in Figure 5 from corresponding properties of the basic components in Figure 1 and the fact that the transformations from the basic components to the final model in Figure 5 considered above are safety properties preserving.

Safety properties say in general that "nothing bad can happen" in a the system. Safety property is expressed by suitable temporal logic formulas. These formulas state a fact about the marking of the net and are given in terms of a number of formal expressions concerning tokens on places. The static formula $2\mathbf{d} \wedge 3\mathbf{a}$ is true for a marking M where at least 2 tokens are present on place **d** and at least 3 tokens on place **a**. The always operator $\Box$ in a safety-property $\Box(2\mathbf{d} \wedge 3\mathbf{a})$ states that static formula $2\mathbf{d} \wedge 3\mathbf{a}$ is true for all reachable markings from M.

In our case study we have the safety property

$$\Box((\mathbf{NOK} \vee \mathbf{OK}) \Longrightarrow \mathbf{W})$$

for the communication unit in the Figure 1c. This property expresses a fact that, independently of the result **NOK** or **OK** of the comparison of the two copies **C1** and **C2** modeled by nondeterministic choice in place **C**, the transmission subunit waits for acknowledgement in place **W**.

For the printer net in Figure 1 we have the safety property that at most one job is processed in each moment, i.e.

$$\Box((\mathbf{P} \text{ xor } \mathbf{PP} \text{ xor } \mathbf{PC}) \wedge \neg(\mathbf{P} \wedge \mathbf{PP} \wedge \mathbf{PC})),$$

where **P**, **PP** and **PC** stand for *Printer*, *Printing prepared* and *Printing completed*, respectively and xor for exclusive or operator.

Surely, we would like to keep these safety properties valid during the transformation. There are two classes of rules available for the application developer

which preserve safety properties for P/T Petri nets, namely transition gluing and place preserving rules, see [GHP99].

The main result concerning safety property preserving transformations reviewed in Section 2.2 states that for each net transformation sequence $N_1 \Longrightarrow^* N_2$ via safety property preserving rules, where N_1 satisfies a safety property φ, we can conclude that net N_2 satisfies a corresponding translated safety property $\mathcal{T}(\varphi)$. In our small case study the rules shown in Figure 2 are *transition gluing* rules and the rule in Figure 4 is *place preserving*. This implies that the two safety properties for the communication unit and the printer considered above are also true in the final model in the Figure 5.

Liveness Preserving Transformations

Let us recall that a net is called live if for each reachable marking m and each transition t there is some other marking m' reachable from m such that t is enabled under m'. We want to show via liveness preserving transformations that the final model in Figure 5 is live (see main result concerning liveness preserving transformations in Section 2.2).

For this purpose we consider the simple model of printing tasks in Figure 6, where it is easy to check directly that this net is live. The printing-refinement and transmission-refinement rules in Figure 7 are liveness preserving rules and can be applied to the net in Figure 6 leading to the final model in Figure 5. (The meaning of marked subnets on the right-hand sides of the rules is explained in Section 2.2.) Since the net in Figure 6 is live and the rules liveness preserving also the final model is live.

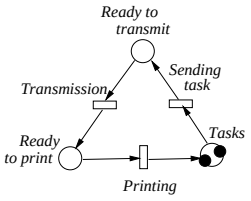
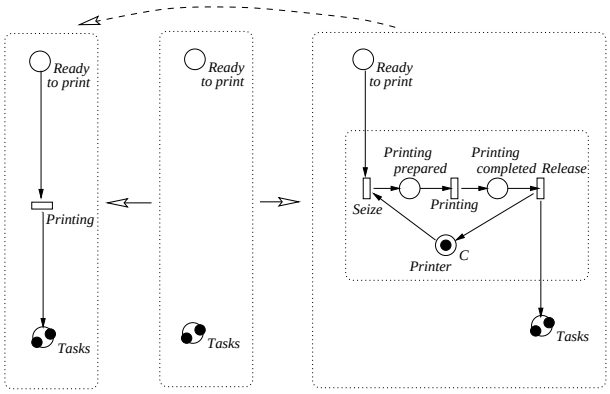


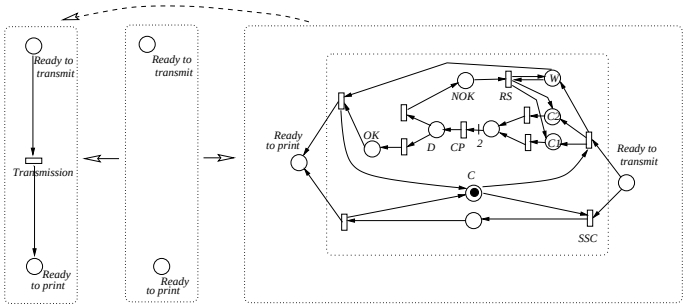
Fig. 6. Simple model of printing tasks

2.2 Expert View

Formal foundation and precise mathematical representation of net model transformations and corresponding results are provided in the Expert View of the ≫Petri Net Baukasten≪. This theory gives a formal categorical framework for stepwise development of systems based on nets, as demonstrated in section 2.1. In following we will present the formal framework of Petri net transformations and main results.



(a) Printing-refinement



(b) Transmission-refinement

Fig. 7. Liveness preserving transition refinement

Formal Framework of Net Model Transformation

The idea of net transformation systems is one of the possible instantiations of the more general idea of high-level transformation systems (see [EHKP91]). In the next sections we summarize the main results. To present all the theoretical results in detail is beyond the scope of the paper. They can be found in the literature cited here.

The next definition introduces rules, transformations and net transformation systems formally for a given category **NET** of low or high level nets. More about the underlying theory can be found in [PER95] and in [Pad99]. We will assume to have a suitable category **NET** of nets and net morphisms, i. e. a category satisfying so called HLR-conditions stated in [EHKP91] and summarized in [PP02] in this volume.

Definition 1 (Rules, Transformations, Net Transformation Systems).

1. *A* rule $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ *in a category* **NET** *consists of the nets* L, K *and* R, *called left-hand side, interface, and right-hand side, respectively, and two morphisms* $K \xrightarrow{l} L$ *and* $K \xrightarrow{r} R$ *with both morphisms* $l, r \in \mathcal{M}$, *a suitable class of injective morphisms in* **NET**.
2. *Given a rule* $p = (L \xleftarrow{l} K \xrightarrow{r} R)$, *a* direct transformation $G \overset{p}{\Longrightarrow} H$ *from a net* G *to a net* H *is given by two pushout diagrams* **(1)** *and* **(2)** *in the category* **NET** *as shown below.*

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & R \\ {\scriptstyle m}\downarrow & \mathbf{(1)} & \downarrow{\scriptstyle k} & \mathbf{(2)} & \downarrow{\scriptstyle n} \\ G & \xleftarrow[g]{} & C & \xrightarrow[h]{} & H \end{array}$$

   The morphisms $L \xrightarrow{m} G$ *and* $R \xrightarrow{n} H$ *are called occurrences of* L *in* G *and* R *in* H, *respectively. By an occurrence of rule* $r = (L \xleftarrow{l} K \xrightarrow{r} R)$ *in a net* G *we mean an occurrence of the left-hand side* L *in* G.
   In fact, the occurrence morphism m *has to satisfy a specific condition, called gluing condition, in order to be able to apply the rule* p *to the net* G.
3. *Given a category of nets* **NET** *together with a suitable class of injective morphisms* $\mathcal{M}$, *a net transformation system* $H = (\mathcal{S}, \mathcal{P})$ *in* $(\mathbf{NET}, \mathcal{M})$ *is given by a start net* $\mathcal{S} \in |\mathbf{NET}|$, *and a set of rules* $\mathcal{P}$.

Although the net transformation framework is a suitable concept for stepwise development of systems, very often there is a need to consider in addition more general morphisms for refinement or abstraction. The main idea is to enlarge the category of nets by $\mathcal{Q}$-morphisms in the sense of [Pad96] in order to formulate refinement/abstraction morphisms.

More precisely, another category of nets **QNET** with a distinguished class of morphisms $\mathcal{Q}$, called $\mathcal{Q}$-morphisms, is employed. The category **QNET** enriches the net transformation system defined in $(\mathbf{NET}, \mathcal{M})$ and yields the notion of $\mathcal{Q}$-morphisms and $\mathcal{Q}$-transformations. The class of $\mathcal{Q}$-morphisms has to satisfy

additional requirements called $\mathcal{Q}$-conditions (see [Pad96]) to be adequate for refinement or abstraction. The formal definitions are given below.

Definition 2 (Class of $\mathcal{Q}$-morphisms). *Let* **QNET** *be a category, so that* **NET** *is a subcategory* $\mathbf{NET} \subseteq \mathbf{QNET}$ *and* $\mathcal{Q}$ *a class of morphisms in* **QNET**.

1. *The morphisms in* $\mathcal{Q}$ *are called* $\mathcal{Q}$*-morphisms, or refinement/abstraction morphisms.*
2. *The* $\mathcal{Q}$*-morphisms must satisfy so called* $\mathcal{Q}$*-conditions, namely closedness of* $\mathcal{Q}$*, preservation of pushouts and inheritance of* $\mathcal{Q}$*-morphisms under pushouts and coproducts.*

Moreover, let $\mathcal{O} = (\mathcal{O}^q)_{q \in \mathcal{Q}}$ be an indexed class of adequate occurrence morphisms in **QNET** with respect to a refinement morphism $q \in \mathcal{Q}$. These occurrence morphisms restrict applicability of rules as defined in Theorem 1 below.

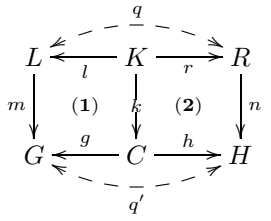
Definition 3 ($\mathcal{Q}$-Rules and $\mathcal{Q}$-Transformations).

1. *A (preserving)* $\mathcal{Q}$*-rule* (p, q) *is given by a rule* $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ *in* **NET** *and a* $\mathcal{Q}$*-morphism* $q : L \to R$*, so that* $q \circ l = r$ *in* **QNET**.
2. *A respecting* $\mathcal{Q}$*-rule* (p, q) *is given by a rule* $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ *in* **NET** *and a* $\mathcal{Q}$*-morphism* $q : R \to L$*, so that* $q \circ r = l$ *in* **QNET**.

The next fact states that $\mathcal{Q}$-morphisms are preserved by transformations.

Theorem 1 (Induced Transformations and Pushouts in QNET). *Let* **QNET** *be a supercategory of* **NET** *according to Definition 2.*

1. *Given a preserving* $\mathcal{Q}$*-rule* (p, q) *and a transformation* $G \overset{p}{\Longrightarrow} H$ *in* **NET** *with an occurrence* $m \in \mathcal{O}^q$ *defined by the pushouts* **(1)** *and* **(2)**, *there is a unique* $q' \in \mathcal{Q}$*, such that* $q' \circ g = h$ *and* $q' \circ m = n \circ q$ *in* **QNET**. *The transformation* $(G \overset{p}{\Longrightarrow} H, q' : G \to H)$*, or* $G \overset{(p,q')}{\Longrightarrow} H$ *for short, is called* $\mathcal{Q}$*-transformation.*
2. *Given a respecting* $\mathcal{Q}$*-rule* (p, q) *and a transformation* $G \overset{p}{\Longrightarrow} H$ *with an occurrence* $n \in \mathcal{O}^q$ *in* **NET** *defined by the pushouts* **(1)** *and* **(2)**, *there is a unique* $q' \in \mathcal{Q}$*, such that* $q' \circ h = g$ *and* $q' \circ n = m \circ q$ *in* **QNET**. *The transformation* $(G \overset{p}{\Longrightarrow} H, q' : H \to G)$*, or* $G \overset{(p,q')}{\Longrightarrow} H$ *for short, is called* $\mathcal{Q}$*-R-transformation.*



The preserving and respecting transformations differ in the direction of $\mathcal{Q}$-morphisms q, q'. The pushout squares **(1)** and **(2)** represent a transformation in the category **NET** (as in Definition 1). The $\mathcal{Q}$-morphism q is a refinement/abstraction morphism in **QNET**. The morphism q' is the induced morphism (according to Theorem 1) which belongs to the class of $\mathcal{Q}$-morphisms in **QNET** as well.

Compatibility with Fusion and Union

We now introduce two basic constructions for high-level structures, both generalizations of the notions union and fusion introduced by [Jen92] for coloured Petri nets. These notions have been reformulated and generalized in the frame of high-level replacement systems [PER95]. We show the compatibility of these notions with $\mathcal{Q}$-transformations. Similarly to the previous section, we assume to have suitable categories **NET** and **QNET** with a class $\mathcal{Q}$.

Several results concerning horizontal structuring for net transformation systems have been adopted from HLR systems. The two basic structuring constructions for nets are union and fusion. Union allows a construction of larger nets from smaller ones with shared subpart, while fusion is a construction which allows to identify distinguished subnets.

Especially the concept of fusion is more general then the concept of fusion of places often cited in the Petri net literature. The fusion introduced in net transformation systems is not restricted to the fusion of places, but covers also fusion of subnets.

Definition 4 (Fusion and Union).

1. *The fusion of two morphisms $f_1, f_2 : F \to G$ between nets F and G is a net G' together with a morphism $g : G \to G'$ defined by the coequalizer $(g : G \to G', G')$ of f_1 and f_2. The fusion is denoted by $G \circ\!\!\longrightarrow G'$ via (F, f_1, f_2, g), or $G \circ\!\!\xrightarrow{F}\!\! G'$ for short.*
2. *The union of a pair of nets (G_1, G_2) via some interface I with the morphisms $i_1 : I \to G_1$ and $i_2 : I \to G_2$ is given by the pushout $G_1 \to G \leftarrow G_2$ of $G_1 \xleftarrow{i_1} I \xrightarrow{i_2} G_2$ and is denoted by $(G_1, G_2) \rightarrowtail\!\!\!\Longrightarrow G$ via (I, i_1, i_2), or $(G_1, G_2) \rightarrowtail\!\!\!\xRightarrow{I} G$ for short.*

In view of software development methodology it is important that horizontal structuring based on fusion and union is compatible with transformations and $\mathcal{Q}$-transformations. This means that – under certain compatibility conditions – the result is the same whether we apply first union/fusion and then transformation or vice versa. For more details about horizontal structuring and transformations see e.g. [Pad96,Pad99].

Theorem 2. *Given a net transformation system $(\mathcal{S}, \mathcal{P})$ in $(\mathbf{NET}, \mathcal{M})$ such that HLR conditions and the compatibility conditions for fusion and union are satisfied.*

1. *Given a fusion* $G \overset{F}{\circ\!\!\longrightarrow} G'$ *and a compatible transformation* $G \overset{p}{\Longrightarrow} H$*, then there is a common net* H'*, obtained by fusion* $H \overset{F}{\circ\!\!\longrightarrow} H'$ *and also by transformation* $G' \overset{p}{\Longrightarrow} H'$*. That means, we have*

$$G \overset{p}{\Longrightarrow} H \overset{F}{\circ\!\!\longrightarrow} H' = G \overset{F}{\circ\!\!\longrightarrow} G' \overset{p}{\Longrightarrow} H.$$

2. *Given a union* $(G_1, G_2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} G$ *and a compatible transformations* $G_i \overset{p_i}{\Longrightarrow} H_i$ *then, there is a common net* H *obtained by the union* $(H_1, H_2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} H$ *and by the transformation* $G \overset{p_1+p_2}{\Longrightarrow} H$ *via the parallel rule* $p_1 + p_2$*, such that we have*

$$(G_1, G_2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} G \overset{p_1+p_2}{\Longrightarrow} H = (G_1, G_2) \overset{(p_1,p_2)}{\Longrightarrow} (H_1, H_2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} H$$

where $(G_1, G_2) \overset{(p_1,p_2)}{\Longrightarrow} (H_1, H_2)$ *denotes the tupling of the separate transformations of* $G_1 \overset{p_1}{\Longrightarrow} H_1$ *and* $G_2 \overset{p_2}{\Longrightarrow} H_2$*.*

3. *If* $\mathcal{Q}$*-conditions are satisfied then above results are valid also for* $\mathcal{Q}$*-transformations and* $\mathcal{Q}$*-R-transformations.*

Main Results Concerning Preservation of Properties

As discussed in Section 2.1 the concept of $\mathcal{Q}$-morphisms is important in order to study whether the transformation of nets is property preserving. The idea of property preserving transformations has been investigated in [GHP99,PG00a] and [PGE98] for safety properties and in [GPU01] for liveness. Safety-properties for Petri nets are stated as propositional logic formulas based on the actual marking of Petri nets. Morphisms preserving safety properties have been investigated for several types of Petri nets, see [GHP99] for P/T Petri nets, [PG00a] for colored Petri nets and [PGE98] for a class of algebraic-high level nets. This class of morphisms has been applied also in a case study of a medical information system in [Pad99] in order to prove relevant properties of the system.

Definition 5 (Formulas, Translations). *Let* $N = (P, T, pre, post, \widehat{m})$ *be a place/transition net, where* P *and* T *are sets of places and transitions respectively,* pre *and* $post$ *are pre- and post-domain functions and* $\widehat{m}$ *is initial marking.*

1. *A static formula* λp *is given for* $\lambda \in \mathbb{N}$ *and* $p \in P$*. The set of all static formulas over* P *is denoted by* $\mathbb{F}$*; static formulas are build up using the logical operators* $\neg$ *and* $\wedge$ *:*

$$\varphi_1 \in \mathbb{F} \Longrightarrow \neg\varphi_1 \in \mathbb{F},$$
$$\varphi_1 \in \mathbb{F}, \varphi_2 \in \mathbb{F} \Longrightarrow \varphi_1 \wedge \varphi_2 \in \mathbb{F}$$

The validity of formulas is given w. r. t. the marking of a net. Let $m \in P^{\oplus}$ *be a marking of* N *then:*

$$\begin{aligned} m \models_N \lambda p \;\; &\textit{iff} \;\; \lambda p \leq m \\ m \models_N \neg\varphi_1 \;\; &\textit{iff} \;\; \neg(m \models_N \varphi_1) \\ m \models_N \varphi_1 \wedge \varphi_2 \;\; &\textit{iff} \;\; (m \models_N \varphi_1) \wedge (m \models_N \varphi_2) \end{aligned}$$

2. *Let φ be a static formula over N. Then $\Box\varphi$ is a safety property. The safety property $\Box\varphi$ holds in N under m iff φ holds in all markings m' reachable from m:*

$$m \models_N \Box\varphi \iff \forall m' \in [m\rangle : m' \models_N \varphi.$$

If m is the initial marking $\widehat{m}$ we also write $N \models \Box\varphi$ instead of $\widehat{m} \models_N \Box\varphi$.

3. *The translation $\mathcal{T}_f$ of formulas over N_1 along a loose morphism $f = (f_P, f_T)$: $N_1 \to N_2$ (see below) to formulas over N_2 is given for atoms by*

$$\mathcal{T}_f(\lambda p) = \lambda f_P(p).$$

The translation of formulas is given recursively by

$$\begin{aligned} \mathcal{T}_f(\neg\varphi) &= \neg\mathcal{T}_f(\varphi), \\ \mathcal{T}_f(\varphi_1 \wedge \varphi_2) &= \mathcal{T}_f(\varphi_1) \wedge \mathcal{T}_f(\varphi_2) \text{ and} \\ \mathcal{T}_f(\Box\varphi) &= \Box\mathcal{T}_f(\varphi). \end{aligned}$$

There have been two notions of safety properties preserving morphisms investigated, namely place preserving and transition gluing morphisms. The formal definition of these morphism is following.

Definition 6 (Place Preserving and Transition Gluing Morphisms). *Given $N_i = (P_i, T_i, pre_i, post_i, \widehat{m_i}), i \in \{1, 2\}$ two place/transition nets. A morphism $f = (f_P, f_T) : N_1 \to N_2$ with functions $f_P : P_1 \to P_2$ and $f_T : T_1 \to T_2$ is called*

loose *if the following embedding conditions hold for all $t \in T_1$ and $p \in P_1$:*

- *(a)* $f_P^{\oplus}(pre_1(t)) \leq pre_2(f_T(t))$ *and* $f_P^{\oplus}(post_1(t)) \leq post_2(f_T(t))$
- *(b)* $f_P^{\oplus}(\widehat{m_1}_{|p}) \leq \widehat{m_2}_{|f_P(p)}$

place preserving *if it is a loose morphism and the following place preserving conditions hold:*

- *(c)* $\bullet(f_P(p)) = f_T^{\oplus}(\bullet p)$ *and* $(f_P(p))\bullet = f_T^{\oplus}(p\bullet)$ *for all* $p \in P_1$
  where $\bullet p = \sum_{t \in T} post(t)(p) \cdot t$ *and* $p\bullet = \sum_{t \in T} pre(t)(p) \cdot t$ *define the pre and post sets of p*
  and $pre(t), post(t) \in P_1^{\oplus}$ *are considered as functions from P to $\mathbb{N}$.*
- *(d)* f_T *and* f_P *are injective*
- *(e)* $\widehat{m_2}_{|f_P} = f_P^{\oplus}(\widehat{m_1})$

transition gluing *if it is a loose morphism and the following holds:*

- *(f)* f_P *is isomorphism*
- *(g)* $f_P^{\oplus}(\widehat{m_1}) = \widehat{m_2}$
- *(h)* f_T *is surjective s.t.* $pre_2(t_2) = \sum_{t_1 \in f_T^{-1}(t_2)} pre_1(t_1)$
  with $t_1 \in T_1$ and $t_2 \in T_2$. post analogously.

Place preserving and transition gluing morphisms yield the safety property preserving transformations as in the next theorem, for which a proof can be found in [GHP99].

Theorem 3 (Safety Property Preserving Transformations). *Given a preserving $\mathcal{Q}$-rule $(p = (L \leftarrow K \rightarrow R), q : L \rightarrow R)$ with q being*

- *either a place preserving morphism*
- *or a transition gluing morphism*

then we have for each $\mathcal{Q}$-transformation step $N_1 \overset{(p,q)}{\Longrightarrow} N_2$ (under appropriate occurrence morphism from $\mathcal{O}^q$) holds:

$$N_1 \models \Box\varphi \Longrightarrow N_2 \models \mathcal{T}_q(\Box\varphi).$$

In our small case study in Section 2.1 in the Figures 2 and 4, the rules and transformations are (according to Theorem 3) safety property preserving, actually the rules in Figure 2 are based on transition gluing morphisms, the rule in Figure 4 is based on a place preserving morphism.

Liveness preserving refinement is based on the standard notion of liveness as used in Petri net theory. We do not consider liveness in the sense of temporal logic formulas as it is often used in litereature. Liveness in our approach means that no deadlock and even livelock can occur. The formal introduction of liveness follows.

Definition 7 (Liveness). *A place/transition net $N = (P, T, pre, post, \widehat{m})$ is called* live *if for arbitrary $m_1 \in [\widehat{m}\rangle$ and arbitrary $t \in T$ there exists some $m_1' \in [m_1\rangle$ such that t is enabled under m_1'.*

In [GPU01] it is shown that a special type of transition refinement preserves liveness in Petri nets. The idea is based on abstracting morphisms, which are closely related to vincinity respecting morphisms (introduced in [DM90]). Abstracting morphisms allow abstracting transitions and places to a single transition. A certain subclass of abstracting morphisms, called collapsing morphisms, allows the description of transition refinement as collapsing of a special subnet (called live in-out cycle) to one transition. The formal introduction of abstracting and collapsing morphisms follows.

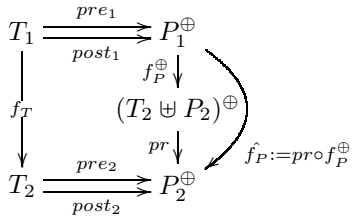
Definition 8 (Abstracting Morphism).

Given two place/transition nets $N_i = (P_i, T_i, pre_i, post_i, \widehat{m_i}), i \in \{1, 2\}$. An abstracting morphism $f : N_1 \rightarrow N_2$ is given by $f = (f_T, f_P)$ with functions $f_T : T_1 \rightarrow T_2$ and $f_P : P_1 \rightarrow (T_2 \uplus P_2)$ such that the following conditions are satisfied:

1. *for all $t \in T_1$:*
   *$f_P(pre_1(t)) = \{f_T(t)\}$ **or** $pr \circ f_P^{\oplus} \circ pre_1(t) = pre_2(f_T(t))$,*
   where $pr : (T_2 \uplus P_2)^{\oplus} \rightarrow P_2^{\oplus}$ is the corresponding projection
   analogously for the post function
2. *for all $t_2 \in f_T(T_1)$:*
   exists $t_{in} \in T_1$ with $f_T(t_{in}) = t_2$ and $pr \circ f_P^{\oplus} \circ pre_1(t_{in}) = pre_2(t_2)$
   analogously for the post function

3. *strict marking:*
   for all $p \in P_1$ *with* $f_P(p) \in P_2$*:* $f_P^{\oplus}(\widehat{m_1}_{|p}) = \widehat{m_2}_{|f_P(p)}$
4. *for all* $p \in P_1$ *with* $f_P(p) \in T_2$*:* $f_T(\bullet p) = \{f_P(p)\}$

analogously for the post function



Example 1. Examples of abstracting morphisms are shown in Figures 7 a,b. The dashed arrow going from the right-hand side of each rule to the left-hand side represents an abstracting morphism between nets. The subnet in a dotted rectangle on the right-hand side of each rule is abstracted by the abstracting morphism to a single transition *Printing*, resp. *Transmission* on the left-hand side.

In order to capture what has been abstracted from a net N_1 by an abstracting morphism, we define the collapsing subnet.

Definition 9 (Collapsing Subnet). *Given an abstracting morphism* $f : N_1 \to N_2$*. We have for all transitions* $t \in T_2$ *the collapsing subnet* $subst_f(t) = (\tilde{P}^t, \tilde{T}^t, \tilde{pre}^t, \tilde{post}^t, \widehat{m_t}) \subseteq N_1$ *with*

- $\tilde{P}^t = \{p_1 \in P_1 \mid f_P(p_1) = t\}$
  these are all places mapped to t
- $\tilde{T}^t = \{t_1 \in T_1 \mid f_T(t_1) = t\}$
  these are all transitions mapped to t
- $\tilde{pre}^t(t_1) = pre_1(t_1)_{|\tilde{P}^t}$ *for all* $t_1 \in \tilde{T}^t$
  the pre- and post-domain restricted to collapsing places
  $\tilde{post}^t$ *is defined analogously.*
- $\widehat{m_t} = \widehat{m_1}_{|\tilde{P}^t}$

Example 2. In Example 1, the subnets in dashed rectangles on the right-hand sides of the rules are exactly the collapsing subnets $subst_f(Printing)$, and $subst_f(Transmission)$ respectively, with respect to the corresponding abstracting morphism f.

Live In-Out Cycles describe those subnets that are live, and are equipped with a guarding place. This guarding place ensures that each run within the subnet has to be completed before it may run again.

Definition 10 (Live In-Out Cycle).
Given a place/transition net $N = (P, T, pre, post, \widehat{m})$*, then* N *is called live in-out cycle if the following conditions hold:*

1. *N is live*
2. *There are two distinguished subsets T_{in} and T_{out} of T, called in-transitions T_{in} and out-transitions T_{out} of N, such that the following holds.*
   There is a safe (1-bounded) place $c \in P$ with $\widehat{m}_{|c} = c$, called guarding place*, which is in the predomain of all in-transitions $t_i \in T_{in}$, and in the post-domain of all out-transitions $t_o \in T_{out}$*
   with $pre(t)_{|c} = \begin{cases} c\,; t \in T_{in} \\ \epsilon\,; t \notin T_{in} \end{cases}$ *and* $post(t)_{|c} = \begin{cases} c\,; t \in T_{out} \\ \epsilon\,; t \notin T_{out} \end{cases}$

Remark 1. The role of the distinguished place c is twofold: on one hand it establishes the cycle, because of condition 2. In this sense it is crucial for liveness in condition 1. On the other hand it implements a mutual exclusion of the in-transitions including the prevention of parallel firing of one transition with itself.

Example 3. The subnets $subst_f(Printing)$ and $subst_f(Transmission)$ in Example 2 are live in-out cycles. The guarding place is named C in both subnets.

Definition 11 (Collapsing Morphisms). *A collapsing morphism $f : N_1 \to N_2$ is an abstracting morphism which additionally satisfies the following conditions. Let*

- $\mathbb{S} \subseteq T_2$ *with* $\mathbb{S} = \{t | \tilde{P}^t \neq \emptyset\}$
- $\mathbb{T} \subseteq T_1$ *with* $\mathbb{T} = \bigcup_{t \in \mathbb{S}} \tilde{T}^t$
- $\mathbb{P} \subseteq P_1$ *with* $\mathbb{P} = \bigcup_{t \in \mathbb{S}} \tilde{P}^t$

then we have

1. *f_T is surjective, and f_P is surjective on $P_1 \setminus \mathbb{P}$*
2. *f_T and f_P are injective on $T_1 \setminus \mathbb{T}$, resp. $P_1 \setminus \mathbb{P}$*
3. *for all $t \in \mathbb{S}$ we have:*
   The collapsing subnet $subst_f(t)$ is a live in-out cycle which is only connected to the rest of N_1 via the in- and out-transitions.

Example 4. The abstracting morphisms in Example 1 are also collapsing morphisms due to the fact that the collapsing subnets are live in-out cycles.

Liveness is respected via collapsing morphisms as shown in [GPU01]. This fact yields liveness preserving transformations as stated below.

Theorem 4 (Liveness Preserving Transformations). *Given a respecting $\mathcal{Q}$-rule $(p = (L \leftarrow K \rightarrow R), q : R \to L)$ with a collapsing morphism q, we have for each $\mathcal{Q}$-R-transformation step $N_1 \overset{(p,q)}{\Longrightarrow} N_2$ (with appropriate occurrence morphism from $\mathcal{O}^q$):*

$$N_1 \quad live \quad \Longrightarrow \quad N_2 \quad live.$$

The rules in our case study in Figure 7 of the Section 2.1 are based on collapsing morphisms. Therefore according to the foregoing theorem we obtain liveness preserving $\mathcal{Q}$-R-transformations. Hence liveness of the simple model in Figure 6 implies liveness of the final model in Figure 5.

3 Net Class Transformations

Numerous kinds of Petri net classes have been introduced in the literature, which allow to apply different Petri net formalisms on different levels of abstraction. In the application developer view we show how net class transformations between different Petri net formalisms can be applied in the development process. In the expert view a formal foundation of net class transformations is given based on the notion of functors in the sense of category theory.

3.1 Application Developer View

Net class transformations can be used by an application developer to begin the modelling in an abstract class with simple nets and to switch to a more expressive class in later development steps.

Moreover, it is important for the development process, that net class transformations are compatible with net model transformations and with the horizontal structuring techniques union and fusion.

In the following we present a development process to derive an algebraic high-level net AHL_4 (Figure 10), via the place/transition net PT_3 (Figure 5 without marking) from the simple elementary net EN_1 (Figure 8) using net model transformations from Section 2.1 and net class transformations $Weight$: **EN** $\rightarrow$ **PT** and $Data$: **PT** $\rightarrow$ **AHL** as shown in Figure 11. In this section we only consider nets without initial marking, but sketch an extension including markings at the end of Sect. 3.2.

We can start the modelling of our running example with the elementary net EN_1 like in Figure 8. Then we can use an elementary net version $printref_{EN}$ of the printing-refinement used in Sect. 2 to elaborate the modelling of the printer. This results in the elementary net EN_2 in Figure 9.

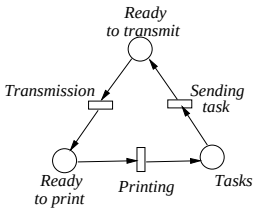


Fig. 8. Simple elementary net EN_1

To use the transmission-refinement $transref_{PT}$ from Sect. 2 as well, we need to transform the model to the class of place/transition nets, because the refinement adds an arc with weight 2. The net class transformation $Weight$ assigns the weight 1 to each arc leading to place/transition nets $PT_1 = Weight(EN_1)$ and $PT_2 = Weight(EN_2)$ and a net model transformation $printref_{PT}$ between

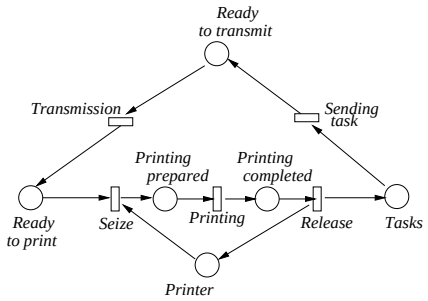


Fig. 9. Refinement of the printer EN_2

them. After the application of the transmission-refinement $transref_{PT}$ we get the same model PT_3 as in Figure 5 of Sect. 2 without marking.

In order to be able to distinguish between different tasks we first transform our model to the class of algebraic high level nets. We obtain algebraic high level nets $AHL_1 = Data(PT_1)$, $AHL_2 = Data(PT_2)$ and $AHL_3 = Data(PT_3)$, where the net class transformation $Data$: **PT** $\rightarrow$ **AHL** adds a trivial data type specification and algebra (see Def. 14 below). The model transformations $printref_{PT}$ and $transref_{PT}$ can be transformed as well yielding transformations $printref_{AHL}$ and $transref_{AHL}$ in Figure 11.

Then we can further refine the net AHL_3 by an AHL net model transformation $specref_{AHL}$, which changes the arc inscriptions of AHL_3 to the ones of AHL_4 given in Figure 10 and adds the following specification with a suitable algebra:

$$
\begin{array}{ll}
 & BOOL+ \\
sorts: & Printer, Task, Comm, Result \\
opns: & printer: \rightarrow Printer \\
 & com: \rightarrow Comm \\
 & task_1: \rightarrow Task \\
 & task_2: \rightarrow Task \\
 & equ: Task\ Task \rightarrow Bool \\
 & ok: Task \rightarrow Result \\
 & nok: \rightarrow Result \\
 & get: Result \rightarrow Task \\
vars: & t: Task, r: Result \\
eqns: & get(ok(t)) = t \\
 & equ(t,t) = true
\end{array}
$$

In the net AHL_4 we are able to model the comparison of the two tasks sent through the insecure channel by inscribing the transition CP with the equation

$$r = \text{if } equ(t', t'') \text{ then } ok(t') \text{ else } nok \ .$$

For simplicity we assume that our non-secure channel either transmits the duplicated task in a correct way (ok-case, $t' = t'' = t$) or t' and t'' are modifications

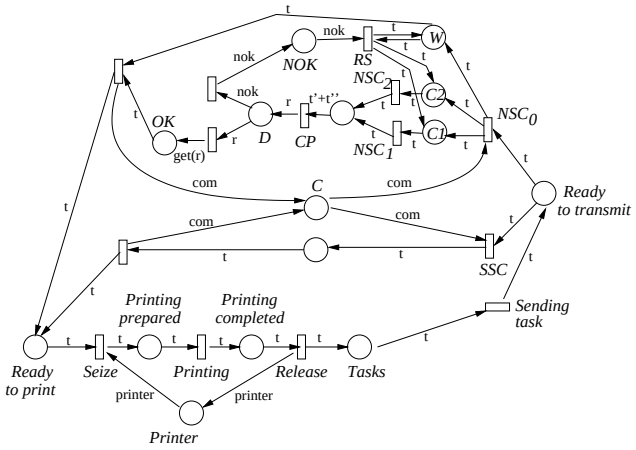


Fig. 10. Final algebraic high level net AHL_4

of t with $t' \neq t''$ (*nok*-case). In the second case the transmission is repeated with the old value t of the task.

The whole development process is depicted in Figure 11, where horizontal steps are net model transformations and vertical steps are net class transformations.

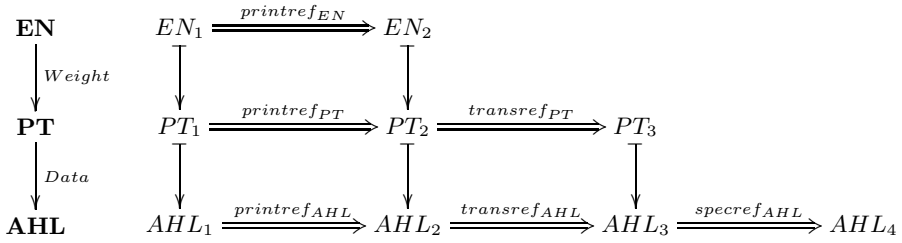


Fig. 11. Development of the printing system

3.2 Expert View

To describe net class transformations as functors we first have to define suitable net morphisms for each net class. In [GPP00,GPP01a] the categories of place/transition nets, elementary nets and algebraic high level nets are introduced in detail, where a net morphism $f = (f_P, f_T)$ is called transition preserving, if f_P and f_T are compatible with the pre- and post-domains of the transitions.

Definition 12 (Categories of Petri Nets).

- *The class of all place/transition nets*

$$PT = (P, T, pre, post) \text{ with } pre, post: T \to P^{\oplus} \ ,$$

where $P^{\oplus}$ is the free commutative monoid over P, together with the transition preserving morphisms

$$f = (f_P, f_T): PT \to PT'$$

constitutes the category of place/transition nets **PT**.
- *The class of all elementary nets*

$$EN = (P, T, pre, post) \text{ with } pre, post: T \to \mathcal{P}(P) \ ,$$

where $\mathcal{P}(P)$ is the powerset of P, together with the transition preserving morphisms

$$f = (f_P, f_T): EN \to EN'$$

constitutes the category of elementary nets **EN**.
- *The class of all marked place/transition nets*

$$mPT = (P, T, pre, post, m_0) \text{ with } pre, post: T \to P^{\oplus}$$
$$\text{and initial marking } m_0 \in P^{\oplus}$$

together with the transition preserving morphisms

$$f = (f_P, f_T): mPT \to mPT' \text{ with } f_P^{\oplus}(m_0) \leq m_0'$$

constitutes the category of marked place/transition nets **mPT**.
- *The class of all algebraic high level nets*

$$AHL = (SPEC, X, A, P, T, pre, post, cond) \ ,$$

where $SPEC = (\Sigma, E)$ is an algebraic specification, A a $SPEC$-algebra and

$$pre, post: T \to (T_\Sigma(X) \times P)^{\oplus} \text{ and } cond: T \to \mathcal{P}(Eqns(\Sigma))$$

are functions, together with transition and equation preserving morphisms

$$f = (f_{SPEC}, f_X, f_A, f_P, f_T): AHL \to AHL'$$

constitutes the category of algebraic high level nets **AHL**.

Definition 13 (Net Class Transformations). *Given categories* **NET** *and* **NET**$'$ *of nets a net class transformation is a functor $T: \mathbf{NET} \to \mathbf{NET}'$.*

We consider the following functors between the categories defined in Def. 12, where for simplicity we omit the transformation of net morphisms. More details are given in [GPP00] and in [PP02] in this volume.

Definition 14 (Functors between Petri Net Categories).

- *Between* **PT** *and* **EN** *there is a functor*

$$Causality\colon \mathbf{PT} \to \mathbf{EN}$$

forgetting all weights in the pre- and post-domains of transitions. Vice versa there is a functor

$$Weight\colon \mathbf{EN} \to \mathbf{PT}$$

assigning the weight 1 to all arcs.
- *Between* **PT** *and* **mPT** *there is a functor*

$$Mark\colon \mathbf{PT} \to \mathbf{mPT}$$

adding an empty initial marking to the net. Vice versa there is a functor

$$Unmark\colon \mathbf{mPT} \to \mathbf{PT}$$

forgetting the initial marking.
- *Between* **PT** *and* **AHL** *there is a functor*

$$Data\colon \mathbf{PT} \to \mathbf{AHL}$$

adding an empty specification, an empty algebra and a set of variables where each arc with weight n is inscribed with the sum of n distinct variables. Vice versa there is a functor

$$Skeleton\colon \mathbf{AHL} \to \mathbf{PT}$$

forgetting specification, variables and algebra and weighting each arc with the number of terms it is inscribed with.

Compatibility Results for Net Class Transformations

As pointed out in Sect. 3.1 it is important, that net class transformations are compatible with the horizontal structuring techniques union and fusion. Because union and fusion are finite colimits from the categorical point of view, this compatibility is achieved by a net class functor preserving colimits.

Definition 15 (Compatibility with Union and Fusion). *A net class transformation $T\colon \mathbf{NET} \to \mathbf{NET}'$ is compatible with union and fusion, if it transforms unions $(N_1, N_2) \overset{I}{\rightarrowtail\!\!\!\!\!\Rightarrow} N$ in* **NET** *into unions $(T(N_1), T(N_2)) \overset{T(I)}{\rightarrowtail\!\!\!\!\!\Rightarrow} T(N)$ in* **NET**′ *and fusions $N \overset{F}{\circ\!\!\longrightarrow} N'$ in* **NET** *into fusions $T(N) \overset{T(F)}{\circ\!\!\longrightarrow} T(N')$ in* **NET**′*, if T preserves colimits.*

Another desirable property is the compatibility of net class transformations with model transformations defined in the previous section. Transformations in one Petri net class shall be translated to transformations in another class by a net class transformation between these classes. This is achieved by net class functors being compatible with the classes $\mathcal{M}$ of morphisms used to define model transformations.

Definition 16 (Compatibility with Net Model Transformations). *A net class transformation* $T\colon \mathbf{NET} \to \mathbf{NET}'$ *is compatible with net model transformations in* $(\mathbf{NET}, \mathcal{M})$ *and* $(\mathbf{NET}', \mathcal{M}')$*, if it transforms net model transformations* $N \overset{p}{\Longrightarrow} N'$ *in* $(\mathbf{NET}, \mathcal{M})$ *to net model transformations* $T(N) \overset{T(p)}{\Longrightarrow} T(N')$ *in* $(\mathbf{NET}', \mathcal{M}')$ *with* $T(p)\colon T(L) \overset{T(l)}{\longleftarrow} T(K) \overset{T(r)}{\longrightarrow} T(R)$ *for* $p\colon L \overset{l}{\longleftarrow} K \overset{r}{\longrightarrow} R$.

A proof of the following theorem is given in [GPP00] using the fact that all the mentioned functors preserve finite colimits.

Theorem 5. *The net class transformations given in Definition 14 are compatible with union, fusion and net model transformations in* $(\mathbf{NET}, \mathcal{M})$ *and* $(\mathbf{NET}', \mathcal{M}')$ *for suitable classes* $\mathcal{M}$ *and* $\mathcal{M}'$ *of injective net morphisms.*

Lifting of Net Class Transformations

In [Pad96] a theory of abstract Petri nets is introduced where different classes of Petri nets are described uniformly by net structure functors $Net\colon \mathbf{Set} \to \mathbf{Set}$ leading to classes **PNC** of nets with $pre, post\colon T \to Net(P)$. For example, the class of place/transition nets **PT** is obtained by the construction of the free commutative monoid $\_^{\oplus}\colon \mathbf{Set} \to \mathbf{Set}$ and the class of elementary nets by the powerset functor $\mathcal{P}\colon \mathbf{Set} \to \mathbf{Set}$.

It can be shown that a natural transformation $v\colon Net \Rightarrow Net'$ between two net structure functors yields a net class functor $F_v\colon \mathbf{PNC} \to \mathbf{PNC}'$. With this fact the functors *Causality* and *Weight* can be reconstructed by giving natural transformations $c\colon \_^{\oplus} \Rightarrow \mathcal{P}$ and $w\colon \mathcal{P} \Rightarrow \_^{\oplus}$ between the free commmutative monoid functor $\_^{\oplus}$ and the powerset functor $\mathcal{P}$.

To integrate algebraic high level nets into this approach pre- and post-domains are replaced by $pre, post\colon T \to Net(T_\Sigma(X) \times P)$. The usual algebraic high level nets **AHL** are obtained by using the net structure functor $\_^{\oplus}\colon \mathbf{Set} \to \mathbf{Set}$. By using $\mathcal{P}\colon \mathbf{Set} \to \mathbf{Set}$ instead we get the new class **EAHL** of elementary high level nets.

The concept of marked nets can be abstracted by taking into account initial markings $m_0 \in Net(P)$ resp. $m_0 \in Net(T_\Sigma(X) \times P)$. This leads to the usual marked place/transition nets **mPT** with $m_0 \in P^{\oplus}$ as well as new classes of marked elementary nets **mEN** with $m_0 \in \mathcal{P}(P)$, marked algebraic high level nets **mAHL** with $m_0 \in (T_\Sigma(X) \times P)^{\oplus}$ and marked elementary high level nets **mEAHL** with $m_0 \in \mathcal{P}(T_\Sigma(X) \times P)$.

By combining these different abstract notions we obtain the cube of Petri net classes and net class transformations given in Fig. 12. More details about the Petri net cube can be found in [GPP00] and in [PP02] in this volume.

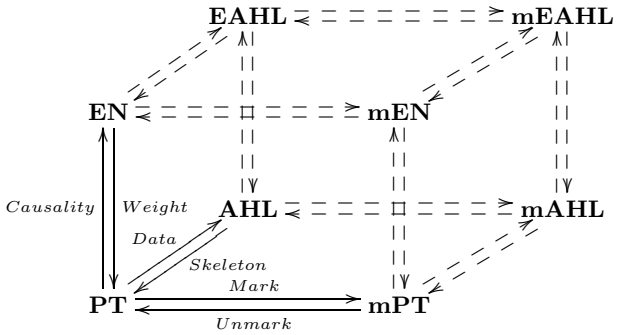


Fig. 12. Cube of Petri net categories

4 Overview of Other Transformations and Conclusion

In this final section we give an overview of other transformations in the »Petri Net Baukasten« and discuss open problems.

4.1 Other Transformations in the the »Petri Net Baukasten«

In addition to net class transformations, there are several transformations between net classes and other modeling techniques which are important for the Application Developer View. One important aspect is to bridge the gap between semiformal techniques in engineering and formal techniques in computer science, which is one of the main aims of the "DFG-Schwerpunktprogramm: Integration von Techniken der Softwarespezifikation für Ingenieurwissenschaftliche Anwendungen". A typical example is the transformation of the MFERT modeling technique for production automation into timed hierarchical Predicate/Transition nets. Another important aspect is to bridge the gap between semiformal and formal techniques in computer science, where visual modeling techniques including message sequence charts (MSC) and different kinds of UML diagrams are considered to be semiformal because no widely accepted formal semantics is available. On the other hand, these visual techniques are very important especially in the phase of requirement analysis. A typical example is the transformation from MSC-scenarios to Petri nets, which has been used in the DFG-SPP reference case study "Railway Traffic Crossing" in order to transform different railway crossing scenarios presented by MSC-diagrams into Petri nets (see [KPE00]).

Finally let us discuss some other kinds of transformations which are important in the Tool Developer View of the »Petri Net Baukasten«. One of the main problems for the integration of different tools has been the different net representation formats for each of these tools. The "Petri Net Markup Language" developed by the "DFG-Forschergruppe Petrinetz-Technologie" provides XML-schemes resp. DTD-standards for different net representation formats. For

n different individual formats we only have to provide 2n explicit transformations from each individual format to the DTD-standard and vice versa in order to provide n(n-1) transformations between each pair of different individual formats:

$$IndividualFormat_i \leftrightarrow DTD\text{-}Standard \leftrightarrow IndividualFormat_j$$

For similar reasons a graphical exchange language GXL is going to be developed in the graph transformation community (see [Tae01]). In order to allow integration of Petri and graph transformation tools it is important to have transformations between the corresponding representation formats PN-DTD for Petri nets and GRA-DTD for graphs. A transformation from PN-DTD to GRA-DTD has been implemented already in [Ehr01], which allows to use graph transformation tools for Petri nets. A typical application is the use of the GenGEd-approach & environment [Bar99] for Visual modeling languages, behavior & animation in order to allow the animation of Petri nets using domain specific visual modeling (see [BEE00]) and the contribution in this volume for more details.

4.2 Conclusion and Open Problems

In this paper we have given an overview of different kinds of Petri net transformations and their relevance for the ≫Petri Net Baukasten≪. Especially we have discussed the use of net model transformations for strepwise development of communication based systems and that of net class transformations for Petri net based software development models. These aspects are important for the Application Developer View of the ≫Petri Net Baukasten≪. The formal framework for net model transformations are double pushout transformations in a suitable category **NET** of Petri nets and that for net class transformations are functors between different categories of Petri nets. These concepts and the corresponding theory are important contributions for the Expert View of the ≫Petri Net Baukasten≪. Finally we have discussed transformations between individual formats of Petri net representations for tools and suitable DTD-standards which are important in the Tool Developer View.

It remains open to integrate the concepts and instances of all these different kinds of transformations into the 2nd Installment of the ≫Petri Net Baukasten≪ concerning user interfaces, services and data bases. Moreover, in each of the areas discussed above there are still several interesting open problems. One typical problem is to study liveness preserving transformations also for algebraic high level nets in section 3.1 similar to those for place/transition nets in section 2.1.

References

[AHS90] J. Adamek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories.* Series in Pure and Applied Mathematics. John Wiley and Sons, 1990.

[Bar99] R. Bardohl. GenGEd – *Visual Definition of Visual Languages Based on Algebraic Graph Transformation.* PhD Thesis, TU Berlin, Verlag Dr. Kovac, Germany, 1999.

[BEE00] R. Bardohl, H. Ehrig, and C. Ermel. Generic Description, Behaviour and Animation of Visual Modeling Languages. In *Proc. Integrated Design and Process Technology*. Dallas, USA, 2000.

[DFG99] DFG-Forschergruppe Petri Net Technology. *Initial Realization of the ≫Petri Net Baukasten≪*. Informatik-Berichte 129, Humboldt-Universität zu Berlin, October 1999.

[DM90] J. Desel and A. Merceron. Vicinity Respecting Net Morphisms. In *Advances in Petri Nets*, pages 165–185. Lecture Notes in Computer Science 483. Springer Verlag, 1990.

[EGP99] H. Ehrig, M. Gajewski, and F. Parisi-Presicce. High-Level Replacement Systems with Applications to Algebraic Specifications and Petri Nets. In *Handbook of Graph Grammars and Computing by Graph Transformation*, Volume 3: Concurrency, Parallelism, and Distribution, pages 341–400. World Scientific, 1999.

[EHKP91] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. From Graph Grammars to high level replacement systems. In *4th Int. Workshop on Graph Grammars and their Application to Computer Science*, pages 269–291. Lecture Notes in Computer Science 532. Springer Verlag, 1991.

[Ehr01] K. Ehrig. Converting XML Files with XSLT and XPATH. http://tfs.cs.tu-berlin.de/lehre/SS01/gragra.html, 2001. Student's Project Status Report.

[Gaj00] M. Gajewsky. Concepts and Requirements for Transformations within Petri Net Based Process Models. In *5th World Conference on Integrated Design and Process Technology*, Special Session on Model Integration. CD-ROM, 8 pages, 2000.

[GE01] M. Gajewsky, and H. Ehrig. The ≫Petri Net Baukasten≪. In *Unifying Petri Nets*, Advances in Petri Nets, Lecture Notes in Computer Science 2128. Springer Verlag, 2001.

[GHP99] M. Gajewsky, K. Hoffmann, and J. Padberg. *Place Preserving and Transition Gluing Morphisms in Rule-Based Refinement of Place/Transition Systems*. Technical Report 1999-14, Technical University Berlin, 1999.

[GPP00] M. Gajewsky, and F. Parisi-Presicce. *Formal Transformations of Petri Nets*. Technical Report 2000-12, Technical University Berlin, 2000.

[GPP01a] M. Gajewsky, and F. Parisi-Presicce. Transformations between Petri Net Classes with Application to Software Development. In *2nd Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems*. Berlin, 2001.

[GPP01b] M. Gajewsky, and F. Parisi-Presicce. On Compatibility of Model and Class Transformations. In *Recent Trends in Algebraic Development Techniques*, 15th International Workshop WADT 2001, pages 24–25, Lecture Notes in Computer Science 2267. Springer Verlag, 2001.

[GPU01] M. Gajewsky, J. Padberg, and M. Urbášek. *Rule-Based Refinement for Place/Transition Systems: Preserving Liveness-Properties*. Technical Report 2001-8, Technical University of Berlin, 2001.

[Jen92] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, Volume 1: Basic Concepts. EATCS Monographs in Theoretical Computer Science 28. Springer Verlag, 1992.

[KPE00] O. Kluge, J. Padberg, and H. Ehrig. Modeling Train Control Systems: From Message Sequence Charts to Petri Nets. In *Proc. Formale Techniken für die Eisenbahnsicherung (FORMS)*, pages 25–42. Fortschritt-Berichte VDI, 2000.

[Lil95] J. Lilius. *On the Structure of High-Level Nets*. PhD thesis, Helsinki University of Technology, 1995. Digital Systems Laboratory, Research Report 33.

[Pad96] J. Padberg. *Abstract Petri Nets: A Uniform Approach and Rule-Based Refinement*. PhD thesis, Technical University Berlin, 1996. Shaker Verlag.

[Pad99] J. Padberg. Categorical Approach to Horizontal Structuring and Refinement of High-Level Replacement Systems. In *Applied Categorical Structures* 7(4), pages 371–403, 1999.

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic High-Level Net Transformation Systems. In *Math. Struct. in Comp. Science*, Volume 5, pages 217–256, 1995.

[Peu01] S. Peuker. *Halbordnungsbasierte Verfeinerung zur Verifikation verteiler Algorithmen*. PhD thesis, Humboldt University Berlin, 2001.

[PG98] J. Padberg, and M. Gajewsky. Using High-Level Replacement Systems to Preserve Safety Properties in Place/Transition Net Transformations. In *Sixth Int. Workshop on Theory and Application of Graph Transformation*, pages 356–365. Universität-Gesamthochschule Paderborn, Fachbereich Mathematik-Informatik, 1998.

[PG00a] J. Padberg, and M. Gajewsky. *Safety Preserving Transformations of Coloured Petri Nets*. Technical Report 2000-13, Technical University Berlin, 2000.

[PG00b] J. Padberg, and M. Gajewsky. Rule-Based Refinement of Petri Nets For Modeling Train Control Systems. In *Petri Nets in Design, Modelling and Simulation of Control Systems*, Special Session at the IFAC Conference on Control Systems Design, pages 299–304, 2000.

[PGE98] J. Padberg, M. Gajewsky, and C. Ermel. Rule-Based Refinement of High-Level Nets Preserving Safety Properties. In *Fundamental Approaches to Software Engineering*, pages 221–238. Lecture Notes in Computer Science 1382. Springer Verlag, 1998.

[PGH99] J. Padberg, M. Gajewsky, and K. Hoffmann. Incremental Development of Safety Properties in Petri Net Transformations. In *Theory and Application of Graph Transformation*, pages 410–425. Lecture Notes in Computer Science 1764. Springer Verlag, 1999.

[PHG00] J. Padberg, K. Hoffmann, and M. Gajewsky. Stepwise Introduction and Preservation of Safety Properties in Algebraic High-Level Net Systems. In *Fundamental Approaches to Software Engineering*, pages 249–265. Lecture Notes in Computer Science 1783. Springer Verlag, 2000.

[PP02] F. Parisi-Presicce. A Formal Framework for Petri Net Class Transformations. In *Petri Net Technology for Communication Based Systems*, Lecture Notes in Computer Science. Springer Verlag, 2002.

[Tae01] G. Taentzer. Towards Common Exchange Formats for Graphs and Graph Transformation Systems. In *Int. Workshop on Uniform Approaches to Graphical Process Specification Techniques (UNIGRA'01)*, Sattelite Event of ETAPS'01, 2001.

[WER99] H. Weber, H. Ehrig, and W. Reisig, editors. *Int. Colloquium on Petri Net Technology for Modelling Communication Based Systems*, Part II: The ≫Petri Net Baukasten≪. Fraunhofer Gesellschaft ISST, October 1999.