Petri Net Transformations

Hartmut Ehrig, Kathrin Hoffmann, Julia Padberg,
Claudia Ermel, Ulrike Prange, Enrico Biermann, Tony Modica
Institute for Software Technology and Theoretical Computer Science
Technical University of Berlin, Germany

August 14, 2007

1 Introduction

Modelling the adaption of a system to a changing environment gets more and more important. Application areas cover e.g. computer supported cooperative work, multi agent systems, dynamic process mining or mobile networks. One approach to combine formal modelling of dynamic systems and controlled model adaption are Petri net transformations. The main idea behind net transformation is the stepwise development of place/transition nets by given rules. Think of these rules as replacement systems where the left-hand side is replaced by the right-hand side while preserving a context. This approach increases the expressiveness of Petri nets and allows in addition to the well known token game a formal description of structural changes.
The chapter is structured as follows: We start with a general overview of net transformations [25, 30, 7, 10] in Section 2.
In Section 3, we illustrate the rule-based refinement of place/transition nets in terms of a case study in the area of an emergency scenario [4]. The case study shows how to use Petri net transformations as refinement concept and demonstrates the compatibility of net refinement and net composition which indicate the relevance of Petri net transformations for software engineering.
In Section 4, we present precise definitions of basic notions concerning Petri net transformations in the case of place/transition nets. The union theorem shows the compatibility of net transformations with the union of nets via a common interface provided that the net transformations are preserving this interface. Furthermore, results for high-level nets are also briefly discussed at the end of Section 4. In the conclusion we discuss how tools for graph transformation systems can also be used for Petri net transformations.

2 General Overview of Net Transformations

The main idea of net transformations is the rule-based modification of nets where each application of a rule leads to a net transformation step. While the well-known token game of Petri nets does not change the net structure, the concept of Petri net transformations is a rule-based approach for dynamic changes of the net structure of Petri nets. Since Petri nets can be considered as bipartite graphs the concept of graph transformations can be applied to define transformations of Petri nets. In the following we give a general overview of graph and net transformations, for more details see [30, 8, 12, 7, 14].

The research area of graph transformation is a discipline of computer science which dates back to the early seventies. Methods, techniques, and results from the area of graph transformation have already been studied and applied in many fields of computer science such as formal language theory, pattern recognition and generation, compiler construction, software engineering, concurrent and distributed systems modelling, database design and theory, logical and functional programming, AI, visual modelling, etc. Graph transformation has at least three different roots, namely from Chomsky grammars on strings to graph grammars, from term rewriting to graph rewriting, and from textual description to visual modelling.

Computing by graph transformation is a fundamental concept for programming, specification, concurrency, distribution, and visual modelling. A state of the art report for applications, languages and tools for graph transformation on the one hand and for concurrency, parallelism and distribution on the other hand is given in volumes 2 and 3 of the *Handbook of Graph Grammars and Computing by Graph Transformation* [8] and [12]. In our paper [14], we have presented a comprehensive presentation of graph and net transformations and their relation. Petri net transformations can also be realized for algebraic high-level nets [25], which is a high-level net concept integrating algebraic specifications with place/transition nets.

In contrast to most applications of the graph transformation approach, where graphs denote states of a system, and rules and transformations describe state changes and the dynamic behavior of systems, in the area of Petri nets we use rules and hence transformations to represent stepwise modification of nets. This kind of transformation for Petri nets is considered to be a vertical structuring technique, known as rule-based net transformation. Basically, a rule (or production) $p = (L, R)$ is a pair of graphs (or nets) called left-hand side L and right-hand side R. Applying the rule $p = (L, R)$ means to find a match of L in the source graph (or net) and to replace L by R. In order to replace L by R we need to connect R with the context leading to the target graph (respectively the target net) of the transformation.

The well-known argument in favour of formal techniques, to have precise notions and rigid mathematical results, clearly holds for this approach as well. Moreover, we have already investigated net transformations in high-level Petri net classes (see Subsection 4.6) that are even more suitable for system modelling than the place/transition nets in our example. The impact for system development is

founded in what results from net transformations:

- *Stepwise Development of Models:* The model of a complex software system may reach a size that is difficult to handle and may compromise the advantages of the (formal) model severely. The one main counter measure is breaking down the model into sub-models, the other is to develop the model top-down. In top-down development the first model is a very abstract view of the system and step by step more modelling details and functionality are added. In general, however, this results in a chain of models, that are strongly related by their intuitive meaning, but not on a formal basis. Petri net transformations fill this gap by supporting the formal step-by-step development of a model. Rules describe the required changes of a model and their applications yield the transformations of the model. Moreover, the representation of changes in a visual way using rules and transformations is very intuitive and does not require a deeper knowledge of the theory.

- *Distributed Development of Models:* Decomposing a large model is an important technique for the development of complex models. To combine the advantages of a horizontal structuring with the advantages of step-by-step development, vertical structuring techniques for ensuring the consistency of the composed model are required. Then a distributed step-by-step development is available that allows the independent development of sub-models. The theory of net transformation comprises horizontal structuring techniques and ensures compatibility between these and the transformations. In Subsection 4.4 we introduce the union construction for the decomposition, and the union theorem in Subsection 4.5 allows to develop the subnets independently of each other. The theory allows complex compositions and decompositions, where the independence of the sub-models is essential. So, the formal foundation for the distributed development of complex models is given.

- *Incremental Verification:* Pure modification of Petri nets is often not sufficient, since the net has some desired properties that have to be ensured during further development. Verification of each intermediate model requires a lot of effort and hence is cost intensive. But refinement can be considered as the modification of nets preserving desired properties. Hence the verification of properties is only required for the net where they can be first expressed. In this way properties are introduced into the development process and are preserved from then on. Rule-based refinement modifies Petri nets using rules and transformations so that specific system properties are preserved. For a brief discussion see Subsection 4.6.

- *Foundation for Tool Support:* A further advantage is the formal foundation of rule-based refinement and/or rule-based modification for the implementation of tool support. Due to the theory of Petri net transformations we have a precise description how rules and transformations work on Petri

nets. Tool support is the main precondition for the practical use. The user should get tool support for defining and applying rules. The tool should assist the choice as well as the execution of rules and transformations.

- *Variations of the Development Process:* Another application area, where transformations are very useful, concerns variations in the development process. Often a development is not entirely unique, but variations of the same development process lead to variations in the desired models and resulting systems. These variations can be expressed by different rules yielding different transformations, that are used during the step-by-step development.

3 Emergency Scenario Case Study

In this section we illustrate the main idea of net transformations by a case study of a pipeline emergency scenario where an unknown source of a natural gas leak is detected in a residential area[1]: A postal worker delivering mail in a residential street smells a strong odor of gas. She immediately notifies the fire department. A single engine company is dispatched by the fire department with four firefighters led by one company officer. At the scene, the postal worker meets the company officer and describes the problem. He calls the gas company and requests additional law enforcement officers to control traffic into the area. While three firefighters evacuate the homes in the immediate area and afterwards deny entry to this area, the forth one reads the gas indicator and detects that the gas is highest in front of a home located on 114 Maple Street. After electricity and gas lines are shut off to each home the fire department people stand by with fully charged hose lines and wait for the arrival of the gas company.
The cooperative process enacted by the firefighter company is depicted as Petri net **PN1** in Fig. 1. This Petri net is decomposed into five parts corresponding to the team members described above, and in addition start as well as end activities. The union describes the gluing of the subnets along the interface given by the post domain places of transition *Start* (respectively pre domain places of transition *End*).
In this case the interface net consists of places only, so that the union corresponds to the usual place fusion of nets. But the general union construction allows having arbitrary subnets as interfaces.
In the following we show how Petri net transformations can be used in the case study before we present the basic concepts in Section 4. The three firefighters responsible for the evacuation process need more detailed information how to proceed. So the company officer gives the instruction that first of all the residents shall be notified of the evacuation. Afterwards the firefighters shall assist handicapped persons and guide all of them to the extent possible. To introduce the refinement of the *Evacuate homes*-transition into the Petri net **PN1** we provide the rule $r_{evacuate}$ depicted in the upper row of Fig. 2.

[1] www.pipelineemergencies.com

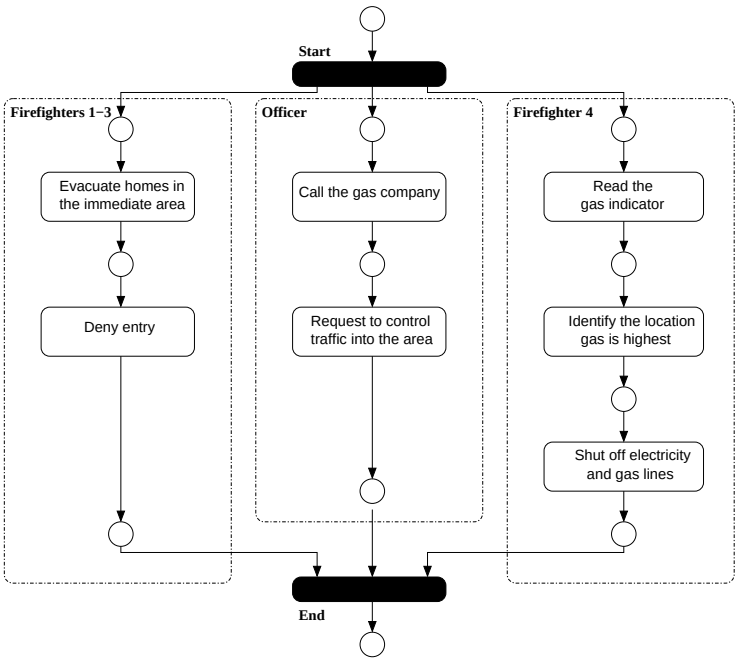


Figure 1: Petri Net **PN1**

We show explicitly the direct transformation with rule $r_{evacuate}$ from **Firefighters 1-3** (see Fig. 1) to **Firefighters 1-3′** in Fig. 2. The application of the rule is given as follows: the match morphism m is given by the obvious inclusion and identifies the relevant parts of the left hand side **L1** of rule $r_{evacuate}$ in **Firefighter 1-3**. In the first step we delete from **Firefighter 1-3** the *Evacuate homes*-transition and adjacent edges, but we preserve all places of **L1**, because they are also in **K1** and **R1**, leading to the context net **C** in Fig. 2. In the second step we glue together **C** and **R1** via **K1** by adding the transitions *Notify residents*, *Assist handicapped persons* and *Guide persons* together with their (new) environment to the context net **C** leading to **Firefighters 1-3′** in Fig. 2. Thus we obtain the direct transformation **Firefighters 1-3** $\overset{r_{evacuate}}{\Longrightarrow}$ **Firefighters 1-3′**.

Since the rule $r_{evacuate}$ and the direct transformation are preserving the interface of the corresponding union in Fig. 1, the interfaces are still available and can be used to construct a resulting net. The union theorem in Section 4 makes sure that this construction leads to the same result as if we would have applied the rule $r_{evacuate}$ to the entire net **PN1** in Fig. 1. This is a typical example for compatibility of horizontal structuring (union) with vertical refinement (rule-based transformation).

After the problem identification the odor of gas grows stronger and the firefighter takes an additional reading of the gas indicator and informs the company officer

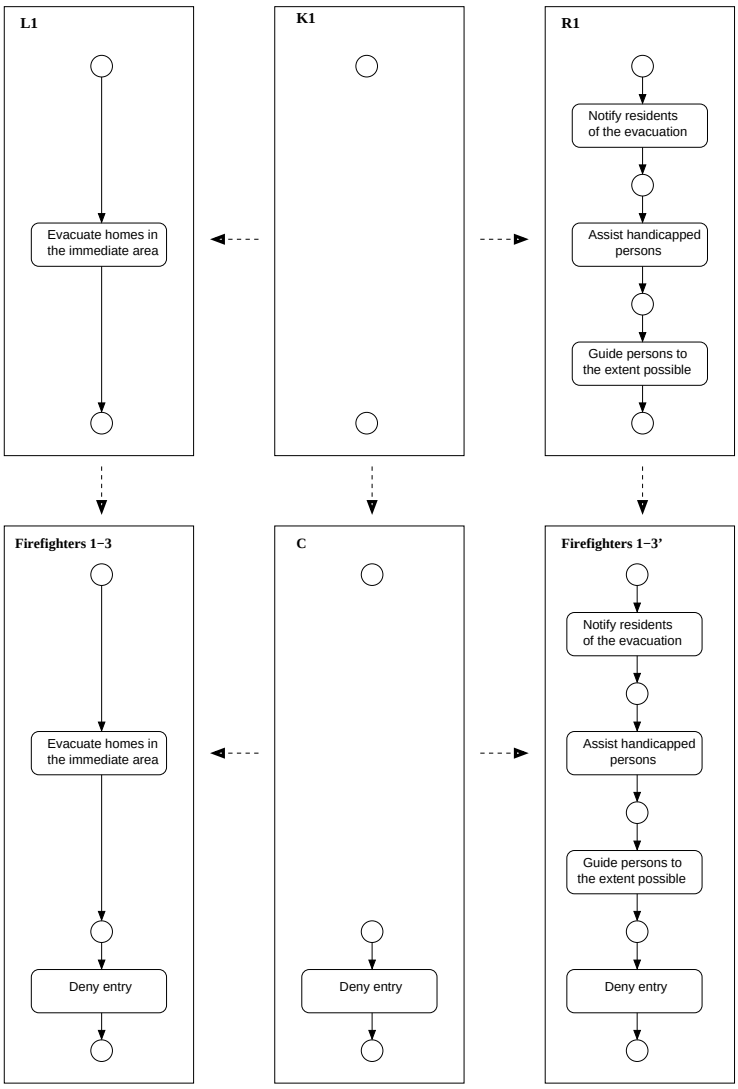


Figure 2: Direct transformation **Firefighters 1-3** $\overset{r_{evacuate}}{\Longrightarrow}$ **Firefighters 1-3′**

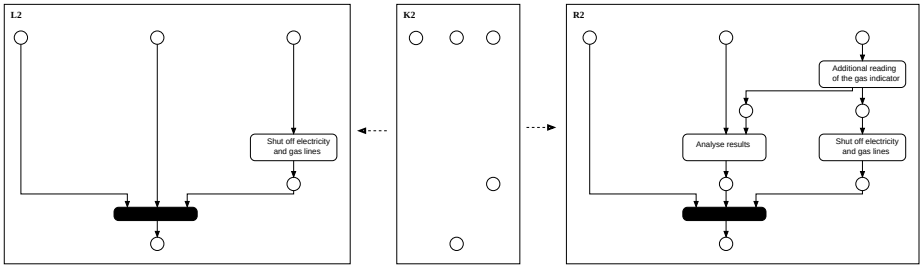


Figure 3: Rule $r_{analyse}$

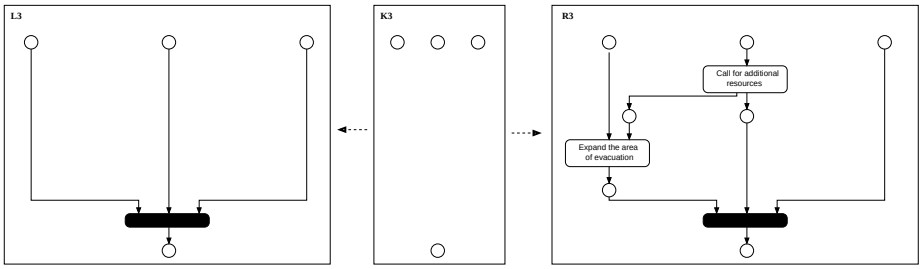


Figure 4: Rule r_{expand}

about the result, so that the company officer is able to determine if the atmosphere in the area is safe, unsafe, or dangerous. To extend our process by these additional activities we use the rule $r_{analyse}$ in Fig. 3.
Based on the additional results of the gas indicator the company officer analyses that the atmosphere in this area is over the lower explosive limit and thereby more dangerous than expected. He determines that the best course of action is to call for additional resources to maintain the isolation perimeter and expand the area of evacuation as a precaution. Here, we use rule r_{expand} depicted in Fig. 4 to extend the Petri net by the additional activities.
Summarizing, after the sequential application of the rules $r_{evacuate}$, $r_{analyse}$ and r_{expand} to the Petri net **PN1** in Fig. 1 we obtain the Petri net **PN4** in Fig. 5.

4 Concepts of Petri Net Transformations

Following up the informal overview in Section 2 we give in this section the precise definitions of the notions that we have already used in our example. For notions and results beyond that we give a brief survey in Subsection 4.6 and refer to literature.
The concept of Petri net transformations is a special case of high-level replacement systems. High-level replacement systems have been introduced in [9] as a categorical generalisation of the double-pushout approach to graph transformation, short DPO-approach. The theory of high-level replacement systems can be

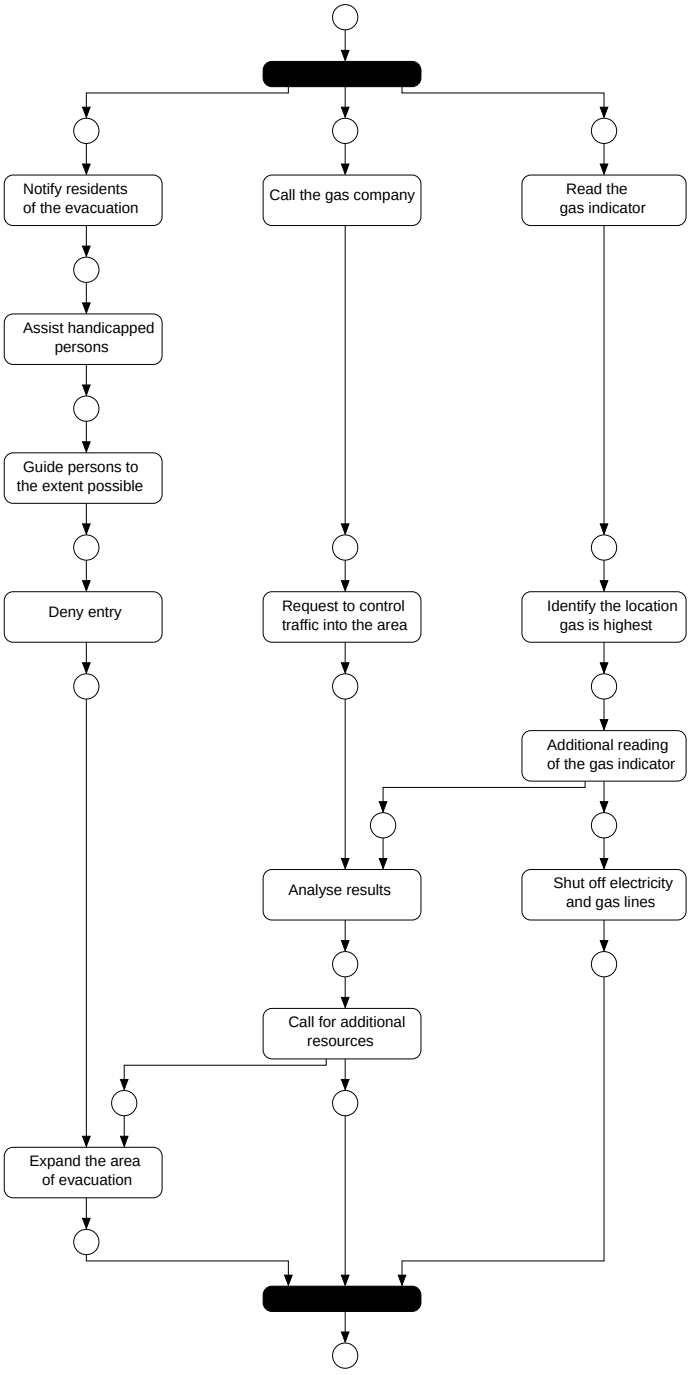


Figure 5: Petri net **PN4**

successfully employed not only to graph transformation, but also to other areas as Petri nets (see [9]). This leads to the concept of Petri net transformations as an instantiation of high-level replacements systems. In the following we explicitly present the resulting concept of Petri net transformations for the case of place/transition nets.

4.1 Place/Transition Nets and Net Morphisms

Let us first present a notation of place/transition net that is suitable for our transformation approach. We assume that the nets are given in the algebraic style as introduced in [21]. A place/transition net $N = (P, T, pre, post)$ is given by the set of places P, the set of transitions T, and two mappings $pre, post : T \to P^{\oplus}$, the pre-domain and the post-domain,

$$T \underset{post}{\overset{pre}{\rightrightarrows}} P^{\oplus} ,$$

where $P^{\oplus}$ is the free commutative monoid over P that can also be considered as the set of finite multisets over P. The pre- (and post-) domain function maps each transition into the free commutative monoid over the set of places, representing the places and the arc weight of the arcs in the pre-domain (respectively in the post-domain). For finite P, an element $w \in P^{\oplus}$ can be presented as a linear sum $w = \sum_{p \in P} \lambda_p \cdot p$ with $\lambda_p \in \mathbb{N}$ or as a function $w : P \to \mathbb{N}$. In the infinite case we have to require that $\lambda_p \neq 0$ only for finitely many $p \in P$ that means the corresponding $w : P \to \mathbb{N}$ has finite support.
In the net **L3** in Fig. 4 T consists of one transition t and four places, where p_1, p_2, p_3 are shown above and p_4 below of t. The function $pre : T \to P^{\oplus}$ and $post : T \to P^{\oplus}$ are defined by $pre(t) = p_1 \oplus p_2 \oplus p_3$ and $post(t) = p_4$, respectively.
Based on the algebraic notion of Petri nets we use simple homomorphisms that are generated over the set of places. These morphisms map places to places and transitions to transitions.
A morphism $f : N_1 \to N_2$ between two place/transition nets $N_1 = (P_1, T_1, pre_1, post_1)$ and $N_2 = (P_2, T_2, pre_2, post_2)$ is given by $f = (f_P, f_T)$ with mappings $f_P : P_1 \to P_2$ and $f_T : T_1 \to T_2$ so that $pre_2 \circ f_T = f_P^{\oplus} \circ pre_1$ and $post_2 \circ f_T = f_P^{\oplus} \circ post_1$. These conditions ensure that the pre-domain as well as the post-domain of a transition are preserved, so that, even if places may be identified the number of tokens that are taken, remains the same. The morphism $f = (f_P, f_T)$ is called injective, if f_P and f_T are injective. The diagram schema for net morphisms is given in the following diagram.

$$\begin{array}{ccc} T_1 & \underset{post_1}{\overset{pre_1}{\rightrightarrows}} & P_1^{\oplus} \\ \downarrow{\scriptstyle f_T} & & \downarrow{\scriptstyle f_P^{\oplus}} \\ T_2 & \underset{post_2}{\overset{pre_2}{\rightrightarrows}} & P_2^{\oplus} \end{array}$$

Several examples of net morphisms can be found in Fig. 2 where the horizontal and vertical arrows denote injective net morphisms.

4.2 Rules and Transformations

The formal definition of rules and transformations is based on concepts of the following category **PT**. The category **PT** consists of place/transition nets as objects and place/transition net morphisms as morphisms. In order to formalise rules and transformations for nets we first state the construction of pushouts in the category **PT** of place/transition nets. For any span of morphisms $N_1 \leftarrow N_0 \rightarrow N_2$ the pushout can be constructed and means intuitively the gluing of nets N_1 and N_2 along N_0. The construction is based on the pushouts for the sets of transitions and sets of places in the category **Set**. In the category **Set** of sets and functions the pushout object D is given by the quotient set $D = B + C/\equiv$, short $D = B +_A C$, where $B + C$ is the disjoint union of B and C and $\equiv$ is the equivalence relation generated by $f(a) \equiv g(a)$ 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 $B + C$ we glue together the elements $f(a) \in B$ and $g(a) \in C$ for each $a \in A$.
Given the morphisms $f : N_0 \rightarrow N_1$ and $g : N_0 \rightarrow N_2$ then the pushout N_3 in the category **PT** with the morphisms $f' : N_2 \rightarrow N_3$ and $g' : N_1 \rightarrow N_3$ is constructed (see diagram below) as follows:

- $T_3 = T_1 +_{T_0} T_2$ with f'_T and g'_T as pushout of f_T and g_T in **Set**.
- $P_3 = P_1 +_{P_0} P_2$ with f'_P and g'_P as pushout of f_P and g_P in **Set** as well.
- $pre_3(t) = \begin{cases} [pre_1(t_1)] & ; \text{ if } g'_T(t_1) = t \\ [pre_2(t_2)] & ; \text{ if } f'_T(t_2) = t \end{cases}$
- $post_3(t) = \begin{cases} [post_1(t_1)] & ; \text{ if } g'_T(t_1) = t \\ [post_2(t_2)] & ; \text{ if } f'_T(t_2) = t \end{cases}$

$$\begin{array}{ccc} N_0 & \xrightarrow{f} & N_1 \\ {\scriptstyle g}\downarrow & (=) & \downarrow{\scriptstyle g'} \\ N_2 & \xrightarrow[f']{} & N_3 \end{array}$$

Two examples of the pushout construction of nets are depicted in Fig. 2. We have the embedding of **K1** into **L1** and **C**. The pushout describes the gluing of the nets **L1** and **C** along the two places of the interface **K1**. Hence we have the pushout $\mathbf{L1} +_{\mathbf{K1}} \mathbf{C} =$**Firefighters 1-3** on the left hand side of Fig. 2. Similarly, we have the pushout $\mathbf{R1} +_{\mathbf{K1}} \mathbf{C} =$**Firefighters 1-3′** on the right hand side of Fig. 2.
Since rule application always involves the construction of two pushouts, we speak of the double-pushout (DPO) approach to graph and net transformation, where transformation rules describe the replacement of the left-hand side net by the right-hand side net in the presence of an interface net.

- A rule $r = (L \xleftarrow{k_1} K \xrightarrow{k_2} R)$ consists of place/transition nets L, K and R, called left-hand side, interface and right-hand side net respectively, and two injective net morphisms $K \xrightarrow{k_1} L$ and $K \xrightarrow{k_2} R$.

- Given a rule $r = (L \xleftarrow{k_1} K \xrightarrow{k_2} R)$, a direct transformation $N_1 \overset{r}{\Longrightarrow} N_2$ from N_1 to N_2 is given by two pushout diagrams (1) and (2) in the following diagram. The morphisms $m : L \to N_1$ and $n : R \to N_2$ are called match and comatch, respectively. The net C is called pushout complement or the context net.

$$\begin{array}{ccccc} L & \xleftarrow{k_1} & K & \xrightarrow{k_2} & R \\ {\scriptstyle m}\downarrow & (\mathbf{1}) & \downarrow{\scriptstyle c} & (\mathbf{2}) & \downarrow{\scriptstyle n} \\ N_1 & \longleftarrow & C & \longrightarrow & N_2 \end{array}$$

The illustration of a transformation can be found for our example in Fig. 2, where the rule $r_{evacuate}$ is applied to the net **Firefighters 1-3** with match m. As explained above, the first pushout denotes the gluing of the nets **L1** and **C** along the net **K1** resulting in the net **Firefighters 1-3**. The second pushout denotes the gluing of the nets **R1** and **C** along the net **K1** resulting in the net **Firefighters 1-3′**.

4.3 Gluing Condition and Context Nets

Given a rule r and a match m as depicted in the diagram above, then we construct in the first step the pushout complement C provided that a suitable gluing condition holds. This leads to the pushout (1) in the diagram above. In the second step we construct the pushout of c and k_2 leading to N_2 and the pushout (2) in the diagram above.
Intuitively the gluing condition makes sure that we can construct a context net C, also called pushout complement, from rule r and match m such that the gluing $C +_K L$ of C and L along K is equal to the net N_1. Formally we have to require that dangling points and identification points are gluing points in the following sense:

Gluing Condition for Nets: $DP \cup IP \subseteq GP$, where the gluing points GP, dangling points DP and the identification points IP of L are defined by

- $GP = k_1(P_K \cup T_K)$,
- $DP = \{p \in P_L | \exists t \in (T_1 \setminus m_T(T_L)) : m_P(p) \in pre_1(t) \oplus post_1(t)\}$, and
- $IP = \{p \in P_L | \exists p' \in P_L : p \neq p' \wedge m_P(p) = m_P(p')\}$
  $\cup \{t \in T_L | \exists t' \in T_L : t \neq t' \wedge m_T(t) = m_T(t')\}$.

Now the pushout complement C is constructed by:

- $P_C = (P_1 \setminus m_P(P_L)) \cup m_P(k_{1P}(P_K))$

- $T_C = (T_1 \setminus m_T(T_L)) \cup m_T(k_{1T}(T_K))$
- $pre_C = pre_{1|T_C}$ and $post_C = post_{1|T_C}$

Note that the pushout complement C leads to the pushout (1) in the diagram above and that it is unique up to isomorphism.
In our example in Section 3, the gluing condition is satisfied in the direct transformation in Fig. 2 since the match is injective and places are not deleted by the rule $r_{evacuate}$. In fact, the dangling points DP of the match in Fig. 2 are given by one place of **L1**, while the gluing points GP consists of all places in **L1**. The set of identification points IP is empty, because the match is injective, hence we have $DP \cup IP \subseteq GP$.

4.4 Union Construction

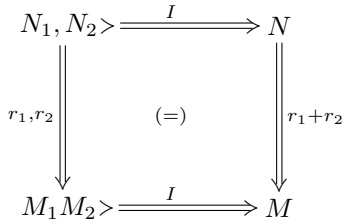
The union of two Petri nets sharing a common subnet, that may be empty, is defined by the pushout construction for nets. The union of place/transition nets N_1, N_2 sharing an interface net I with the net morphisms $f : I \to N_1$ and $g : I \to N_2$ is given by the pushout diagram (1) below. Subsequently we use the short notation $N = N_1 +_I N_2$ or $N_1, N_2 > \stackrel{I}{\Longrightarrow} N$.

$$\begin{array}{ccc} I & \xrightarrow{f} & N_1 \\ {\scriptstyle g}\downarrow & (1) & \downarrow{\scriptstyle g'} \\ N_2 & \xrightarrow[f']{} & N \end{array}$$

In our example in Fig. 1 we can use the union construction several times to describe the net **PN1** as the composition of five different subnets given by **Firefighters 1-3**, **Officer**, **Firefighter 4**, **Start** and **End** . The interface nets I are given by the intersection of the of the corresponding nets.

4.5 Union Theorem

The Union Theorem states the compatibility of union and net transformations in the following sense: A union of two nets followed of a parallel transformation of the united nets yields the same result as two transformations of the original two nets followed by a union of the two transformed nets.
Given a union $N_1 +_I N_2 = N$ and net transformations $N_1 \stackrel{r_1}{\Longrightarrow} M_1$ and $N_2 \stackrel{r_2}{\Longrightarrow} M_2$ then we have a parallel rule $r_1 + r_2 = (L_1 + L_2 \leftarrow K_1 + K_2 \rightarrow R_1 + R_2)$, where $L_1 + L_2$, $K_1 + K_2$ and $R_1 + R_2$ are disjoint unions of the respective nets of rules r_1 and r_2, and a parallel net transformation $N \stackrel{r_1+r_2}{\Longrightarrow} M$. Then $M = M_1 +_I M_2$ is the union of M_1 and M_2 with the shared interface I, provided that the given net transformations preserve the interface I. The Union Theorem is illustrated in the following diagram and especially stated and proven in [22]:



Note that the compatibility requires an independence condition stating that nothing from the interface net I may be deleted by one of the transformations of the subnets.
This allows intersections to apply either the rules $r_1 = r_{evacuate}$ and $r_2 = r_{analyse}$, respectively, to $N_1 =$**Firefighters 1-3** in Fig. 1 and N_2 constructed as union in four steps of the nets **Officer**, **Firefighter 4**, **Start** and **End**, or in parallel to the union $N = N_1 +_I N_2$, where I consists of two places which are preserved by both transformations $N_1 \stackrel{r_1}{\Longrightarrow} M_1$ and $N_2 \stackrel{r_2}{\Longrightarrow} M_2$. This allows to obtain the same net M by union $M = M_1 +_I M_2$ and by transformation $N \stackrel{r_1+r_2}{\Longrightarrow} M$. Finally, applying rule $r_3 = r_{expand}$ to M leads to the net **PN4** in Fig. 5.

4.6 Further Results

We briefly introduce the main net classes which have been studied up to now and subsequently present some main results.

- Place/transition nets in the algebraic style have already been introduced in Subsection 4.1. In [11, 17, 10] we have transferred these results to place/transition systems, where a place/transition system is a place/transition net with an initial marking.
- Coloured Petri nets [18, 19, 20] are high-level nets combining P/T nets and ML expressions for data type definitions. They are very popular due to the tool CPN-tolls [5].
- Algebraic high-level nets are available in quite a few different notions e.g. [28, 25]. We use a notion that reflects the paradigm of abstract data types into signature and algebra. An algebraic high-level net (as in [25]) is given by $N = (SPEC, P, T, pre, post, cond, A)$, where $SPEC = (S, OP, E; X)$ is an algebraic specification in the sense of [13] with additional variables X not occurring in E, P is the set of places, T is the set of transitions, $pre, post : T \to (T_{OP}(X) \times P)^{\oplus}$ are the pre- and post-domain mappings, $cond : T \to \mathcal{P}_{fin}(EQNS(SIG, X))$ are the transition guards, and A is a $SPEC$ algebra.

Horizontal Structuring Union and fusion are two categorical structuring constructions for place/transition nets that merge two subnets (fusion) or two different nets (union) into one.

The union has been introduced in the previous subsection. Now let us consider the fusion: Given a net F that occurs in two copies in the net N_1, represented by two morphisms $F \overset{f}{\underset{f'}{\rightrightarrows}} N_1$, the fusion construction leads to a net N_2, where both occurrences of F in N_1 are merged. If F consists of places $p_1, .., p_n$ then each of the places occurs twice in net N_1, namely as $f(p_1), ..., f(p_n)$, and $f'(p_1), ..., f'(p_n)$. N_2 is obtained from the net N_1 by fusing both occurrences $f(p_i)$ and $f'(p_i)$ of each place p_i for $1 \leq i \leq n$.
The Union Theorem has been presented in the previous subsection. The Fusion Theorem [23] is expressed similarly: Given a rule r and a fusion $F \rightrightarrows N_1$ then we obtain the same result whether we derive first $N_1 \overset{r}{\Longrightarrow} N_1'$ and then construct the fusion $F \rightrightarrows N_1'$ resulting in N_2' or whether we construct the fusion $F \rightrightarrows N_1$ first, resulting in N_2 and then perform the transformation step $N_2 \overset{r}{\Longrightarrow} N_2'$. Similar to the Union Theorem, a certain independence condition is required. Both theorems state that Petri net transformations are compatible with the corresponding structuring technique under suitable independence conditions. In short these conditions guarantee that the interface net I and respectively the fusion net F are preserved by all net transformations.

Interleaving and Parallelism We are able to realize model interleaving and parallelism of net transformations. The Local Church-Rosser Theorem states a local confluence in the sense of formal languages corresponding to interleaving. The required condition of parallel independence means that the matches of both rules overlap only in parts that are not deleted. Sequential independence means that those parts created or used by the first transformation step are not used or deleted in the second step, respectively. The Parallelism Theorem states that sequential or parallel independent transformations can be carried out either in arbitrary sequential order or in parallel. In the context of step-by-step development these theorems are important as they provide conditions for the independent development of different parts or views of the system. More details on horizontal structuring or parallelism are given in [25] and [23].

Refinement Rule-based refinement comprises the transformation of Petri nets using rules while preserving certain net properties. For Petri nets the desired properties of the net model can be expressed e.g in terms of Petri nets (as liveness, boundedness etc.), in terms of logic (e.g. temporal logic, logic of actions etc.), in terms of relation to other models (e.g. bisimulation, correctness etc.), and so on.
For place/transition nets, algebraic high-level nets and Coloured Petri nets the most important results for rule-based refinement are presented in Table 1. For more details see [27].

Notion/Results	PT-nets	AHL-nets	CPNs
Rules, Transformations	√	√	√
Safety property preserving transformations with			
transition-gluing morphisms	√	√	√
place-preserving morphisms	√	√	√
Safety property introducing transformations	√	√	√
Liveness preserving transformations	√	?	?
Liveness introducing transformations	√	?	?
Local Church Rosser I + II Theorem	√	√	√
Parallelism Theorem	√	√	√
Union	√	√	√
Fusion	√	√	√
Union Theorem	√	√	√
Fusion Theorem	√	√	√

Table 1: Achieved results

5 Conclusion

The main idea of Petri net transformations is to extend the classical theory of Petri nets by a rule-based technique that allows to model the changes of the Petri net structure.
There have been already a few approaches to describe transformations of Petri nets formally (e.g. in [2, 3, 31, 6, 32]). The intention has been mainly on reduction of nets to support verification, and not on the software development process as in our case. This use of transformations has been one of the main focus areas of the DFG-Research group *Petri Net Technology*. There are some large studies in various application areas as medical information systems [15], train control systems [26], or as sketched in this paper in emergency scenarios. These case studies clearly show the advantages using net transformation in system development and the practical use of the results stated in Table 1.
Although the area of Petri net transformations is already well-established, there are many promising directions for further research to follow, for example:

- Transfer to other net classes

There is a large variety of Petri net classes, and in principle the idea of Petri net transformation is applicable to all of them. The concept of transformation we have employed is an algebraic one, so the use of algebraic approaches to Petri nets is more suggesting. Algebraic higher-order Nets [16] have been recently developed and are one of the promising targets to transfer the idea of transformations to. These nets extend algebraic high-level nets as they are equipped with a higher-order signature and algebra. This allows most interesting applications and supports structure flexibility and system adaptability in an extensive way.

- Reconfigurable place/transitions systems
  In [17], the concept of reconfigurable place/transition (P/T) systems has been introduced that is most important to model changes of the net structure while the system is kept running. In detail, a reconfigurable P/T-system consists of a P/T-system and a set of rules, so that not only the follower marking can be computed but also the structure can be changed by rule application to obtain a new P/T-system that is more appropriate with respect to some requirements of the environment. Moreover these activities can be interleaved. In [11] we have continued our work on by transferring the results of local Church-Rosser which are well known for term rewriting and graph and net transformations (see [30, 7, 10]), to the consecutive evolution of a P/T-system by token firing and rule applications. In more detail, we assume that a given P/T-system represents a certain system state. The next evolution step can be obtained not only by token firing, but also by the application of one of the rules available. Hence, we have presented conditions for (co-)parallel and sequential independence, such that each of these evolution steps can be postponed after the realization of the other, yielding the same result and, analogously, they can be performed in a different order without changing the result.

- Component technology
  Components present an advanced paradigm for the structuring of complex systems and have been advocated in the recent years most strongly. Components that use Petri nets for the specification of the interfaces and the component body have been defined in [24]. There are three nets that represent the import, the export and the body of the component. The export is an abstraction of the body and the import is embedded into the body. There are two operations: the hierarchical composition and the union of components. Unfortunately, up to now there is no transformation concept in the sense of graph and net transformation. Based on net transformations the transformation of the import, the export and the body can be defined straightforward.

- Tool support
  The practical use of graph transformation is supported by several tools. The algebraic approach to graph transformation is especially supported by the graph transformation environment AGG (see [1]). A tool for net

transformations using the graph transformation engine AGG has been developed recently [29] as a plug-in Eclipse to support a special class of reconfigurable P/T- systems.

References

[1] AGG Homepage. http://tfs.cs.tu-berlin.de/agg.

[2] G. Berthelot. Checking Properties of Nets using Transformations. In *Advances in Petri Nets*, volume 222 of *LNCS*, pages 19–40. Springer, 1986.

[3] G. Berthelot. Transformations and Decompositions of Nets. In *Advances in Petri Nets*, volume 254 of *LNCS*, pages 359–576. Springer, 1987.

[4] P. Bottoni, F. De Rosa, K. Hoffmann, and M. Mecella. Applying Algebraic Approaches for Modeling Workflows and their Transformations in Mobile Networks. *Mobile Information Systems*, 2(1):51–76, 2006.

[5] CPN Tools Homepage. http://wiki.daimi.au.dk/cpntools/_home.wiki.

[6] R. David and H. Alla, editors. *Petri Nets and Grafcet*. Prentice Hall (UK), 1992.

[7] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science. Springer, 2006.

[8] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 2: Applications, Languages and Tools*. World Scientific, 1999.

[9] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and concurrency in high-level replacement systems. *Math. Struct. in Comp. Science*, 1:361–404, 1991.

[10] H. Ehrig, K. Hoffmann, U. Prange, and J. Padberg. Formal Foundation for the Reconfiguration of Nets. Technical Report Technical Report 2007-02, Technical University Berlin, Fak. IV, 2007.

[11] H. Ehrig, J. Padberg K. Hoffmann, U. Prange, and C. Ermel. Independence of Net Transformations and Token Firing in Reconfigurable Place/Transition Systems. In *Proc. Application and Theory of Petri Nets (ATPN)*, 2007.

[12] H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation. Vol 3: Concurrency, Parallelism and Distribution*. World Scientific, 1999.

[13] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*. EATCS Monographs on Theoretical Computer Science. Springer, 1985.

[14] H. Ehrig and J. Padberg. Graph Grammars and Petri Net Transformations. In *Lectures on Concurrency and Petri Nets, Special Issue Advanced Course PNT*, volume 3098 of *LNCS*, pages 496–536. Springer, 2004.

[15] C. Ermel, J. Padberg, and H. Ehrig. Requirements Engineering of a Medical Information System Using Rule-Based Refinement of Petri Nets. In *Proc. Integrated Design and Process Technology (IDPT)*, volume 1, pages 186–193. Society for Design and Process Science, 1996.

[16] K. Hoffmann. *Formal Approach and Applications of Algebraic Higher Order Nets*. PhD thesis, Technical University Berlin, 2005.

[17] K. Hoffmann, H. Ehrig, and T. Mossakowski. High-Level Nets with Nets and Rules as Tokens. In *Proc. Application and Theory of Petri Nets (ATPN)*, volume 3536 of *LNCS*, pages 268–288. Springer, 2005.

[18] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1: Basic Concepts. of *EATCS Monographs in Theoretical Computer Science*. Springer, 1992.

[19] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, volume 2: Analysis Methods of *EATCS Monographs in Theoretical Computer Science*. Springer, 1995.

[20] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, volume 3: Practical Use of *EATCS Monographs in Theoretical Computer Science*. Springer, 1997.

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

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

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

[24] J. Padberg. Basic Ideas for Transformations of Specification Architectures. In *Proc. Workshop on Software Evolution through Transformations (SET 02)*, volume 74 of *ENTCS*, 2002.

[25] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic High-Level Net Transformation Systems. *Mathematical Structures in Computer Science*, 5(2):217–256, 1995.

[26] J. Padberg, P. Schiller, and H. Ehrig. New Concepts for High-Level Petri Nets in the Application Domain of Train Control. In *Proc. Symposium on Transportation Systems*, pages 153–160, 2000.

[27] J. Padberg and M. Urbášek. Rule-Based Refinement of Petri Nets: A Survey. In *Proc. Petri Net Technology for Communication-Based Systems*, volume 2472 of *LNCS*, pages 161–196. Springer, 2003.

[28] W. Reisig. Petri Nets and Algebraic Specifications. *Theoretical Computer Science*, 80:1–34, 1991.

[29] RON Editor Homepage. http://tfs.cs.tu-berlin.de/roneditor/.

[30] G. Rozenberg. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997.

[31] Vanio M. Savi and Xiaolan Xie. Liveness and Boundedness Analysis for Petri Nets with Event Graph Modules. In *Proc. Application and Theory of Petri Nets (ATPN)*, volume 254 of *LNCS*, pages 328–347. Springer, 1992.

[32] W.M.P. van der Aalst. Verification of workflow nets. In *Application and Theory of Petri Nets*, volume 1248 of *LNCS*, pages 407–426. Springer, 1997.