Rule-Based Refinement of Petri Nets: A Survey*

J. Padberg and M. Urbášek

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

Abstract. This contribution provides a thorough survey of our work on rule-based refinement. Rule-based refinement comprises the transformation of Petri nets using rules while preserving certain system properties. Petri net rules and transformations are expressed by morphisms and pushouts. This allows an abstract formulation of our notions independent of a specific Petri net class, as place/transition nets, elementary nets, predicate/transition nets etc. Hence, it is adequate to consider our approach as rule-based refinement of Petri nets in general. We have presented various results in recent years at different conferences. So this contribution gives an overview of our work in a compact form leaving out the technical details.

1 Introduction

Our work on rule-based refinement of Petri nets has been part of the research project "DFG- Forschergruppe PETRINETZ-TECHNOLOGIE". Here we present our results concerning Petri net transformations that preserve system properties. These properties comprise safety properties as well as liveness of Petri nets. The formal foundation is expressed in a categorical way in order to achieve an approach that is valid for different net classes. We illustrate the concepts at length with several examples in different net classes. Here we concentrate on the intuitive notions and the meaning of the main results. As the underlying category theory is based on morphisms we introduce the basic definitions of the involved morphisms and of the rules as well as the transformations.

First in Subsection 1.1 a motivation for Petri net transformations is given stating the advantages of this specific approach as well as relating it to the main work of the "DFG- Forschergruppe PETRINETZ-TECHNOLOGIE". In Subsection 1.2 we discuss the basic concepts of transformation of Petri nets and the way properties of nets can be preserved. As the last part of the introduction we relate our work to literature. Subsequently we present two extensive examples

* This work is 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).

H. Ehrig et al. (Eds.): Petri Net Technology ... , LNCS 2472, pp. 161–196, 2003.
© Springer-Verlag Berlin Heidelberg 2003

in Section 2. The one for place/transition (in Subsection 2.1) nets shows how a token ring is refined preserving certain safety properties as well as liveness. The second example (in Subsection 2.2) concerns the refinement of coloured Petri nets using a simplified example from the context of ETCS (European Train Control Systems). Since the main technical contents is based on morphisms, in Section 3 we give a brief outline of the involved morphism classes both for place/transition nets (in Subsection 3.1) and coloured Petri nets (in Subsection 3.2). Then in Section 4 we state the results. These results are independent of a specific net class, so we need not to differentiate between place/transition nets and coloured Petri nets. We give a short summary and discuss future work in Section 5.

1.1 Motivation

We focus on the specification process of a system in contrast to approaches concerning the verification of (distributed) algorithms as for example in [Peu02] in this book.

Stepwise modification of Petri nets has been an issue as long as they have been used for specification purposes. The need to develop a model in several steps arises as soon as large models are needed for the specification of a system. Our approach - as it is called - is based on rules and provides a visual way of expressing the development steps of a net. The transformation of a net is achieved by applying a rule to the given net. Representing changes as rules in a visual form is very intuitive and does not require a deep understanding of the mathematical theory behind it. In particular we claim that the examples in Section 2 can be followed and even worked out without the knowledge of the underlying theory. But modification alone is often not sufficient. The model describes some desired system properties that need to be guaranteed after each development step. Verification of each intermediate model requires a lot of effort and hence is cost intensive. Obviously the idea of refinement concerns the modification of nets so that system properties are preserved. Hence the verification of those properties needs only to be done once when introduced. Rule-based refinement modifies Petri nets using rules so that specific system properties are preserved. Preservation of system properties by a transformation is to be understood in the following way: If a Petri net has a certain system property (e.g. safety properties, liveness, etc.) then the transformed net has the corresponding property as well. Preservation of system properties is of interest in many applications as it allows omitting the tedious verification of system properties at different stages of the development. Our transformations are based on the double-pushout approach as first introduced for graphs. Rules are given as a span of two Petri net morphisms and its application is achieved by two pushouts (see the following Subsection 1.2); hence the name double-pushout approach.

The notion of Petri net techniques has been developed in the research project "DFG- Forschergruppe Petrinetz-Technologie" and is inspired by the need to have a more application-oriented presentation of Petri nets. The definition of Petri net techniques has been one of the main research areas in this project (see the corresponding papers in these proceedings [WER99, WER01] and [ERW02]

in this book). The concept of Petri net techniques focuses on Petri nets as a specification technique, hence it comprises more than the Petri net model. It may include structuring and refinement of nets, tool support or exchange formats, or process model and exemplary methodology, so that adequate and scalable use of nets is provided for specific application domains (for more details see [ERW02]). The formal Petri net technique is the formal description of Petri net technique. Within a formal Petri net technique one important way to manipulate nets is rule-based refinement. One motivation for Petri net techniques has been the need for generalizing various notions of Petri nets. Many concepts of Petri nets are given for specific net classes, e.g. place/transition nets, elementary nets, algebraic high-level nets, predicate/transition nets, etc. only. Nevertheless, in principle they often are independent of the specific net class. So we have chosen a generic description method. Our approach to rule-based refinement is independent of specific net classes as it is formulated in a more abstract way. We employ category theory that allows coping with objects (i.e. nets) and morphisms. We characterize specific properties a category (i.e. a net class) has to provide and achieve our results on this abstract basis. In fact, the results do not hold for Petri nets only but can be transferred to other specification techniques as state charts, algebraic specifications, graphs etc. We have ensured that rule-based refinement fits into the concept of a Petri net technique. One of the main features of a technique is that the offered possibilities to cope with nets of that specific class are compatible with each other. Rule-based refinement is compatible with various structuring techniques as union and fusion (see [9]). Furthermore process models [7, 5] based on rule-based refinement are provided.

1.2 Basic Ideas

Rules and transformations of Petri nets are given by an instantiation of high-level replacement systems. These can be considered as general description of replacement systems, where a left-hand side of the rule is replaced by a right-hand side. Generally, rules and transformations fully capture the replacement and thus can define any kind of system development or modification. High-level replacement systems have been introduced in [3] as a categorical generalization of graph transformations. The application of high-level replacement systems to different domains as place/transition nets, algebraic specification [1] etc. requires a suitable category. High-level replacement systems are formulated for an arbitrary category **CAT** with a distinguished class $\mathcal{M}$ of morphisms called $\mathcal{M}$-morphisms. Here we give the corresponding notions of replacement systems in terms of Petri nets and not on the abstract level as in [9]. In this paper we have focused on place/transition and coloured Petri nets. The same approach can be applied to other net classes, e.g. algebraic high-level nets [13].

We now explain the used notions rather informally to show that the complex category theory behind it can be omitted in applications. The reader interested in details of the theory is referred to the corresponding papers. The most important notion from category theory used almost throughout whole theory of high-level replacement systems is the notion of a pushout. In the next paragraph we

describe the pushout construction. For illustration see also the Figure 1. Next we briefly address other necessary notions as rules, transformations, property preserving rules, structuring techniques and proof rules.

Pushouts of Petri nets. Pushouts are a categorical construction that requires a commutative square and has some universal properties. Informally, a pushout can be characterized as the "largest" object, that yields a commutative square for two given morphisms without "new" nodes. A pushouts of Petri nets can be considered as a union of nets with respect to a common interface. That is, for injective morphisms we glue the nets together as illustrated in Figure 1. For coloured Petri nets the construction of pushouts has to take additional components like e. g. the data, arc inscription, transition guards, etc. into account. Basically the construction of these components is analogous to the presented construction of the net structure.

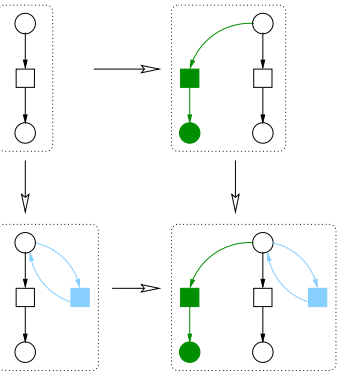


Fig. 1. An example of a pushout

Rules. A rule $r = (L \xleftarrow{k_1} K \xrightarrow{k_2} R)$ consists of the Petri nets L, K and R, called left-hand side, interface and right-hand side respectively, and two $\mathcal{M}$-morphisms $K \xrightarrow{k_1} L$ and $K \xrightarrow{k_2} R$.

Transformations. 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 the following two pushout diagrams **(1)** and **(2)**. The morphisms $L \longrightarrow N_1$ and $R \longrightarrow N_2$ are called occurrences. The net C is called pushout complement.

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

Informally, a rule $r = (L \xleftarrow{k_1} K \xrightarrow{k_2} R)$ is given by three nets L, K, and R. Moreover, K is a subnet of both L and R expressed by the morphisms k_1 and k_2. Application of a rule to net N_1 is a transformation of N_1. The transformation means replacing a subnet specified by the left-hand side of the rule with the net specified by the right-hand side. More precisely, we first identify the subnet L in N_1. Then we delete those parts of the subnet L which are not subnets of the interface net K as well. This results in an intermediate net C, where in a further step we add the difference of R and K to the preserved subnet C to obtain the transformed net N_2. In case the left-hand side is empty, we simply add the right-hand side to the first net. A transformation sequence $N_1 \overset{*}{\Longrightarrow} N_{n+1}$ between nets N_1 and N_{n+1} means that there is a sequence of $n \geq 1$ direct transformations: $N_1 \overset{r_1}{\Longrightarrow} N_2 \overset{r_2}{\Longrightarrow} \ldots \overset{r_n}{\Longrightarrow} N_{n+1}$. In this case we also denote the sequence as $N_1 \overset{r_1, r_2 \ldots r_n}{\Longrightarrow} N_{n+1}$.

Structuring. There are two abstract structuring constructions in the theory of high-level replacement systems, namely union and fusion. Generally, they combine two subnets or two different nets into one. The union of two Petri nets is given with respect to a defined subnet. Union is defined as the pushout of two nets and is given by a span of morphisms. The resulting net preserves the common subnet, i.e. the source of both morphisms and keeps the rest of the two nets distinct, e.g. see Figure 1. The fusion is the gluing of two subnets within one Petri net.

Refinement. Based on the notion of rules and transformations, the general theory of high-level replacement systems has been enriched by the $\mathcal{Q}$-theory in order to formulate abstraction/refinement morphisms of structures These morphisms are more suitable for the stepwise development of of systems. The main idea is to add an abstraction/refinement morphism to a rule going from left-hand side of a rule to the right-hand side or vice versa (see the drawing in the next paragraph). The main advantage of this approach is the fact that the additional abstraction/refinement morphisms can be defined as preserving or reflecting certain properties. This means that certain important system properties may be preserved by transformations as defined below. The general theory of rules and transformations with additional refinement morphisms has been introduced in [9] in the general framework of high-level replacement systems.

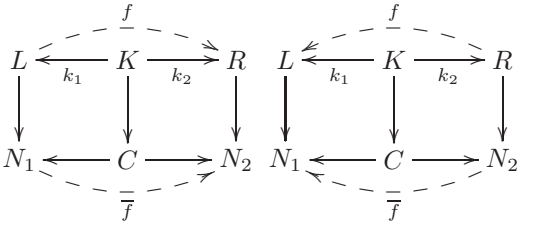
System Properties. Petri nets are an adequate specification technique for behavioral aspects of a system. So, the desired properties of the system to be specified usually concern the behavior of the model. These properties can be expressed in various ways, e.g in terms of Petri nets (as liveness, boundedness 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. Up to now we have focused on liveness of Petri nets and on safety properties in the sense of temporal logic. Liveness of nets means that no deadlock and even livelock of a net can occur, i.e. there always exists a firing sequence which enables any chosen transition from any reachable marking. A safety property is expressed by a logic formula stating facts about markings of a net. A formula is given in terms of numbers of tokens on places. For a place/transition net the static formula $2\mathbf{d} \wedge 3\mathbf{a}$ is true for a marking m where at least 2 tokens are present on the place $\mathbf{d}$ and at least 3 tokens on the place $\mathbf{a}$. The always operator $\Box$ in a safety property $\Box(2\mathbf{d} \wedge 3\mathbf{a})$ requires that the static formula $(2\mathbf{d} \wedge 3\mathbf{a})$ is true for all reachable markings from m. The adaptation of safety properties to coloured Petri nets has to be made due to the different structure of the marking.

Property Preserving Rules. A pair (r, f) is a property preserving rule, if $r = (L \xleftarrow{k_1} K \xrightarrow{k_2} R)$ is a rule with morphisms $k_1, k_2 \in \mathcal{M}$ and with

- either a property preserving morphism $f : L \dashrightarrow R$ s.t. $f \circ k_1 = k_2$.
- or a property respecting morphism $f : R \dashrightarrow L$ s.t. $f \circ k_2 = k_1$.

According to the notion of property preserving morphisms and rules, we can now define property preserving transformations. The general idea is that the application of a rule that preserves properties leads to a net transformation that also preserves these properties.

Property Preserving Transformations. Given a property preserving rule $(r = (L \leftarrow K \rightarrow R), f)$ with $f : L \rightarrow R$ being a property preserving morphism ($f : R \rightarrow L$ being property respecting, respectively). Then the direct transformation $N_1 \overset{(r,f)}{\Longrightarrow} N_2$ is a property preserving transformation[1] with a property preserving (respecting, respectively) morphism $\overline{f} : N_1 \dashrightarrow N_2$ ($\overline{f} : N_2 \dashrightarrow N_1$, respectively). The graphical representation of such transformations is depicted below.



Property preserving transformations represent a strong theoretical result with an obvious impact to the applications. The interpretation of the results is usually stated in form of so-called proof rules. The proof rule indicates the property (stated under the line) which can be derived for a system when certain assumptions (stated above the line) are fulfilled. For the property preserving transformations $N_1 \Longrightarrow N_2$ we have the following proof rule:

$$\frac{(r, f) \text{ is a property preserving rule; } N_1 \text{ satisfies the corresponding property}}{N_2 \text{ satisfies the property too}}$$

The following property introducing transformations are a special kind of general transformations with the empty set on the left-hand side of the rule.

Property Introducing Rules and Transformations. Given a rule $r = (\emptyset \leftarrow \emptyset \rightarrow R)$ such that R satisfies certain property, then the rule is called property introducing rule yielding property introducing transformations $N_1 \Longrightarrow N_2$ and the following proof rule.

$$\frac{r = (\emptyset \leftarrow \emptyset \rightarrow R) \text{ is a rule; } R, N_1 \text{ satisfy a certain property}^{2}}{N_2 \text{ satisfies the property too}}$$

[1] provided the morphisms satisfy certain assumptions

[2] sometimes this condition can be relaxed in such a way that there is no assumption on N_1 made

The proof rules are a powerful expression of the properties of transformations. They are very useful for designers as they abstract from this theory and focus on applications only. Therefore the complicated theory of high-level replacement systems is applicable in practice.

1.3 Related Work

As the reader may already have noted we have in this paper two ways of referring to literature. We have split the references, by having listed our papers on rule-based refinement separately and by referring to other publications as usual. References of our papers on rule-based refinement are given by numbers, the other as an abbreviation of the authors names.

Intuitively, a safety property expressed by a temporal logic formula in the sense of [MP92] means that "nothing bad" can happen in the system, and is thus important for correctness. The combination of transformations with the preservation of safety properties is relevant in software engineering as the verification of such properties in large and complex systems is often necessary, but very complicated (if possible at all) and thus expensive. Thus, it is desirable to state and prove safety properties at an early state and to preserve them throughout the subsequent development. In the area of low-level Petri nets there are many contributions concerning verification with temporal logic [DDGJ90, BS90, HRH91] and refinement [BGV91], [DM90, GG90, BDH92]. In the area of high-level nets, verification [Jen94, Sch96] is much more difficult and even more the compatibility of system properties with refinement. One main problem of verification in formal software engineering can be described by the following demand: Rigorous software development requires continuous verification during all phases of the software development process. Nevertheless, resources are restricted and an entirely new verification at each step is usually considered to be too expensive and time consuming. Thus, vertical structuring techniques should preserve verified properties. Refinement of nets in a way that liveness is preserved is another thoroughly investigated area of Petri nets. Liveness for Petri nets states the fact that in a net all transitions can fire from all reachable markings. Nevertheless, the combination with stepwise modification of nets based on replacement rules has been investigated in [8, 16] only. This approach is in line with previous publications concerning rule-based modification preserving safety properties of nets, see e.g. [13, 15, 14]. A related idea has been pursued in [DA92], where various reduction methods for ordinary nets have been proposed. These methods operate on the net structure and also preserve liveness, by basically shortening paths in the net. The analysis in this case is based on a Petri net reduction. One disadvantage is that the whole Petri net has to be developed entirely and only after the design of a Petri net is finished the analysis can start. Similar remarks hold for reductions presented by other authors in [ES91, Esp94, CT90, FWL]. Authors also use reductions algorithms to decrease the size of the net. J. Esparza, etc. in [ES91, Esp94] investigated the class of free-choice Petri nets, J. Favrel in [FWL] is focused on coloured Petri nets based on the results for generalized Petri nets proposed earlier. In [CT90] the certain subclass of Petri nets, called

regular blocks, is investigated. A work on synthesis methods preserving liveness properties has also been made. In the area of free-choice Petri nets the synthesis preserving liveness properties was described along the reduction methods by J. Esparza, etc. in [ES91, Esp94]. Similar ideas not only for P/T Petri nets can be found in [BGV91, Sou91]. In the area of workflow modeling the notion of soundness which comprises liveness has turned out to be of special importance. In a row of papers, see e. g. [vdA97, vdA98, vdAtH98], v. d. Aalst has stated important results concerning composition of sound nets [vdA98].

In cited papers, the synthesis and/or reduction rules are depicted on more intuitive level. Although our approach is very close to these papers, formalization and net class make an important difference.

2 Examples for Rule-Based Refinement of Petri Nets

In order to illustrate the concepts of rule-based refinement described in Section 1.2 we will present stepwise development of two small systems in this section. The first example is a development of a token ring based on place/transition nets. The another one is a development of a train control system based on coloured Petri nets. A short version of this one can be found in [11]. Together with the development of the systems we gradually introduce system properties as liveness or safety properties. In further development steps these system properties are preserved and thus do not have to be proven again. The process of developing the system is technically based on rules and transformations in the sense of [2]. More formal introduction of used notions and morphisms follows in Section 3.

2.1 Place/Transtion Nets: Token Ring

A token ring is a classical example of a communication protocol between several computational units (CU) on a shared medium. All computational units are connected such that they form a cycle. There is a so-called token sent from one unit to another. The unit which possesses the token can decide either to pass the token through to the next unit or to keep the token and send a message. The sent message propagates through the medium from one unit to another reaching the sending unit sooner or later. Before the sending unit can pass the token to the next unit, the sent data has to be removed from the medium. When a unit receives a data, it retransmit a message to the next unit.

In order to model the token ring system with rule based refinement, the usage of place/transition nets as a basic class of Petri nets is sufficient. The necessary conditions this system has to have are:

1. There exists a unit which possesses a token.
2. The token cannot be doubled, i.e. only one unit has a token.
3. The system is live (without livelocks and deadlocks).

The first proposal of a token ring system with three computational units is given in the Figure 2. This proposal is really a very coarse model of a system.

Nevertheless, it can be refined to get a model which is more suitable. The three conditions of the system stated above can be expressed in this model as follows.

1. $\Box(\mathbf{CU_1} \vee \mathbf{CU_2} \vee \mathbf{CU_3})$
2. $\Box((\mathbf{CU_1} \Rightarrow \neg\mathbf{CU_2} \wedge \neg\mathbf{CU_3}) \wedge (\mathbf{CU_2} \Rightarrow \neg\mathbf{CU_1} \wedge \neg\mathbf{CU_3}) \wedge$
   $\wedge (\mathbf{CU_3} \Rightarrow \neg\mathbf{CU_1} \wedge \neg\mathbf{CU_2}))$
3. The net is live.

The first two conditions can be shortened by using the exclusive-or operator (non-equivalence) as: $\Box((\mathbf{CU_1}$ xor $\mathbf{CU_2}$ xor $\mathbf{CU_3}) \wedge \neg(\mathbf{CU_1} \wedge \mathbf{CU_2} \wedge \mathbf{CU_3}))$.

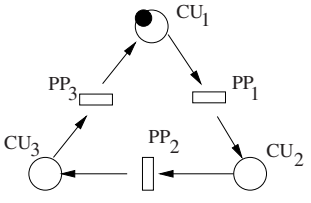


Fig. 2. Token ring - first proposal

Modeling within a rule-based framework starts with a special kind of a rule, which introduces a first proposal of the model. In our example such a rule looks like the one in the Figure 3. This rule is a so-called si- and li-rule, short for safety introducing and liveness introducing rule, respectively (for description see Section 3.1).

The rule introduces a system, which is live and has a safety- property: $\Box((\mathbf{CU_1}$ xor $\mathbf{CU_2}$ xor $\mathbf{CU_3}) \wedge \neg(\mathbf{CU_1} \wedge \mathbf{CU_2} \wedge \mathbf{CU_3}))$. The dashed arrow from the left-hand side to the right-hand side expresses the existence of a place preserving morphism between these nets (as formally introduced in Section 3.1).

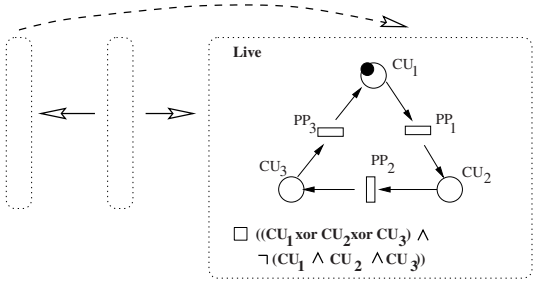


Fig. 3. Introducing the token ring

The proposal of the model does not express the functionality of the computational units. It only expresses the fact that there is a cycle with three units which are connected together. To model full functionality, the first coarse model must be decomposed. The decomposition is done via transformations based on rules. The modeled system has a safety property and is live. We would like to preserve the properties during transformations and omit the check of these properties for final system. We employ so-called liveness preserving rules, which also translate safety properties in some sense. The special kind of transition refinement rules called lp-rules are used (see Section 3.1). We can then perform three transition refinements, one for each unit $i = 1...3$, as sketched in the Figure 4. The dashed arrow means that there exists a collapsing morphism going from the right-hand side net to the left-hand side one in the rule. Surely, in the case of the refinement of $\mathbf{PP_1}$ both places $\mathbf{CU_1}$ in the corresponding rule (for $i = 1$) are marked.

Explanation of used abbreviations:

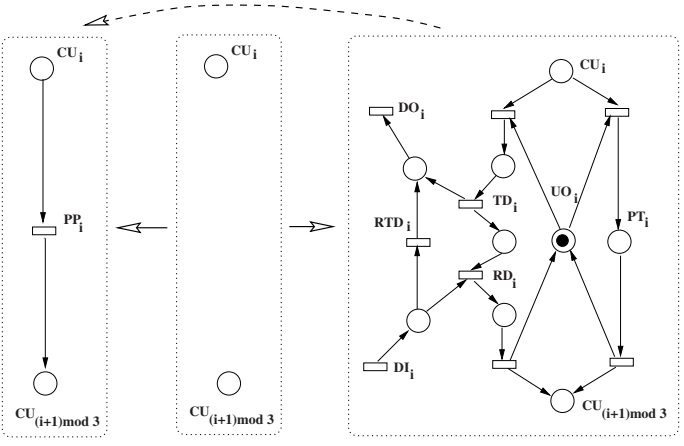


Fig. 4. Transition refinements

PT ... passing a token to next unit, no transmission of data
TD ... transmitting data
RD ... removing data from the medium
RTD... retransmitting data received via medium
UO ... unit not operating
DI ... receiving data
DO ... sending data

The system after refinement is depicted in the Figure 5. Performed transition refinements preserve liveness. The safety property is transformed to:

$$\Box(((\mathbf{CU_1} \vee \neg\mathbf{UO_1}) \text{ xor } (\mathbf{CU_2} \vee \neg\mathbf{UO_2}) \text{ xor } (\mathbf{CU_3} \vee \neg\mathbf{UO_3})) \wedge \neg((\mathbf{CU_1} \vee \neg\mathbf{UO_1}) \wedge (\mathbf{CU_2} \vee \neg\mathbf{UO_2}) \wedge (\mathbf{CU_3} \vee \neg\mathbf{UO_3})))$$

as lp-rules preserve liveness and transform safety properties (as formally explained in the Section 3.1). The transformation of the safety property expresses the fact that when certain computational unit $\mathbf{CU_i}$ possessed the token in the simple original system then an adequate expression after refinement is that either a token is on the place $\mathbf{CU_i}$ or the corresponding added subnet (refinement of $\mathbf{PP_i}$) is running, i.e. there is no token on the place $\mathbf{UO_i}$. The last step in our example is to model a shared medium for data transmission by connecting of apropriate transitions in the Figure 5. For this reason it is necessary to glue a transition DI_i of one unit with a transition $DO_{(i+1)mod\ 3}$ of the next unit. The gluing can be obtained using three l-transition gluing rules as sketched in the Figure 6.

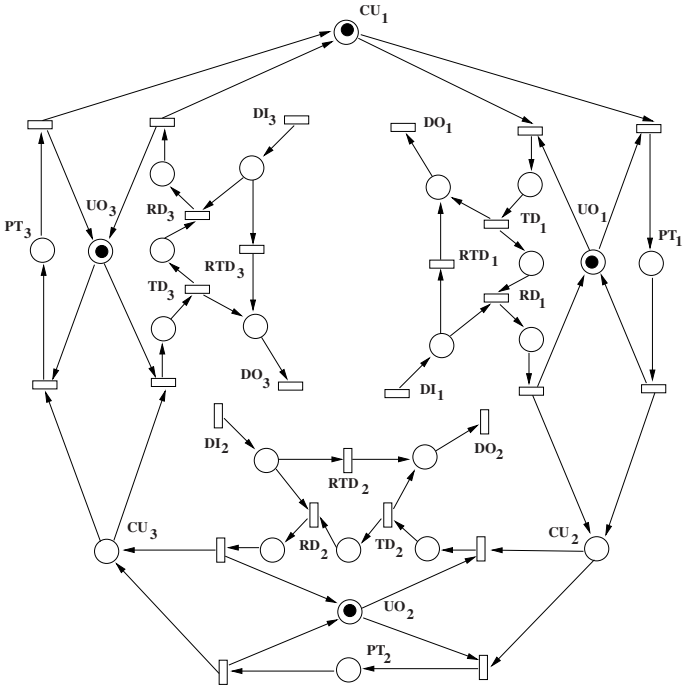


Fig. 5. Refined system

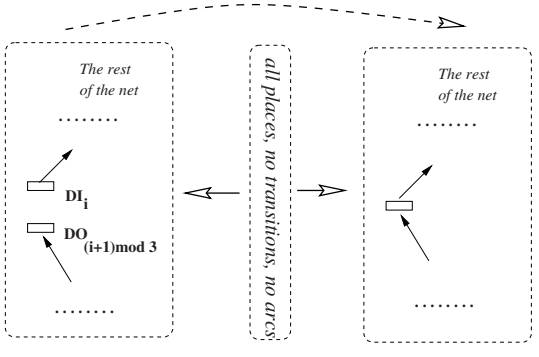


Fig. 6. Transition gluing

The dashed arrows expresses the fact that there exists a l-gluing morphism (and therefore also plain gluing morphism) going from the net on the left-hand side to the one on the right-hand side. In order to simplify the figure, the transitions after gluing are without labels. The final system then is shown in the Figure 7. Liveness and safety properties are preserved via l-gluing morphisms. So, the model developer does not have to check liveness and safety-property for the final model. These properties have been preserved (transformed) during the transformation process and are therefore valid for the final net. Surely, this may save time and effort needed to check system properties, especially in the case of large and complex systems. This approach was also used for rule-based modeling of a large medical information system as presented in [4, 9] . The model developer can, of course, continue in the detailed specification of the model, if necessary.

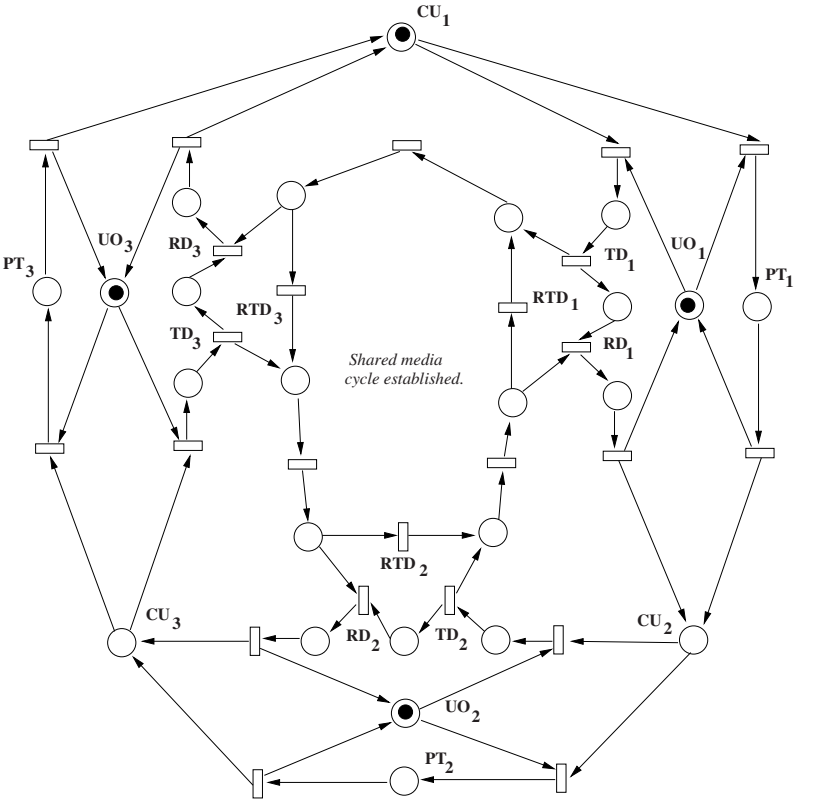


Fig. 7. Final system

2.2 Coloured Petri Nets: Stepwise Development of a Train Control System

High-level nets can be considered as the integration of process and data type description. Some of the most prominent classes are coloured Petri nets [Jen92, Jen94, Jen97], Predicate/Transition nets [GL81, Gen91] and algebraic high-level nets [Vau87, Rei91, 10]. Coloured Petri nets are the focus of this subsection, since they are widely known and constitute a very popular class of high-level Petri nets. The practical relevance of coloured Petri nets is considered to be very high, not at least due to the successful tool Design/CPN [JCHH91]. The presented example concerning train control has been inspired by a case study of a level crossing in a large research project[3] [Ehr97, Jan97], but is of course incomparable with respect to complexity.

[3] DFG-Schwerpunktprogramm "Integration von Techniken der Softwarespezifikation für Ingenieurwissenschaftliche Anwendungen", see also WWW-page `http://tfs.cs.tu-berlin.de/projekte/indspec/SPP/`

Simple Model In our example we develop a coloured Petri net concerning a train's passage of a level crossing, which is one of several situations the train has to meet while traveling to its destination. Figure 8 depicts a simple coloured Petri net, called CPN_1, of the train's behavior in this situation. The used data enclosed in the upper right corner are

1. The train's data consisting of its identity (integer), the distance to a critical section of the track (DISTANCE), and some other data (string) like the schedule or the number of wagons which are not relevant for this example.
2. The distance data is a positive interval of integers, where the lower limit denotes the minimal distance at which the train has to start braking in order to come to a full stop at the critical section. The upper limit is considered to be infinite represented by the value w.
3. The state of the critical section, resp. level crossing, being either safe or unsafe.
4. A black token for requesting the state.

Initially, the train travels along with no knowledge of the next critical section. When it approaches a level crossing, the remaining distance is set by firing of the transition ***approach to level crossing*** in the Figure 8. Consequently, the train is alarmed and requests the state of the level crossing by the transition ***request state of level crossing***. The level crossing state can be either safe or unsafe modeled by the arc inscription of the transition ***set state***. Until the state of the level crossing is set, the train awaits the result. There are three possibilities of action: The train can still come nearer to the level crossing modeled by ***coming nearer to level crossing*** decreasing the distance to the level crossing; or it has already reached the point at which it must begin to brake in which case the transition ***braking*** is enabled. Or — provided the state of the level crossing is safe — it can proceed regularly by the transition ***proceed*** resulting in its passage of the level crossing.
In case of braking, it comes to a full stop before the level crossing. The train has to wait until the state of the level crossing is made safe (by the driver). Subsequently, the train may pass. Finally, it exits the level crossing returning to its initial state. Together with the net there is given a safety property, depicted at the bottom line of the Figure 8. For security reasons it should always be guaranteed that the state of the level crossing is safe when the train passes. This fact is expressed as an appropriate logic formula. Let $\alpha \in C(\textbf{passing level crossing}) = \text{TRAIN}$ be a variable of the colour of the place **passing level crossing**. The safety property $\Box(\alpha, \textbf{passing level crossing}) \Longrightarrow (\text{safe}, \textbf{state of crossing})$ in the Figure 8 means that "In any case, if a train is passing the level crossing, the level crossing is safe". And it is satisfied for the initial marking based on the initialization function. Intuitively, we can argue as follows: The formula $(\alpha, \textbf{passing level crossing}) \Longrightarrow (\text{safe}, \textbf{state of crossing})$ is satisfied in the initial state. Moreover, the two transitions ***proceed*** and ***making l.c. safe*** are the only ones to put a token on **passing level crossing**, which makes the premise of the formula true.

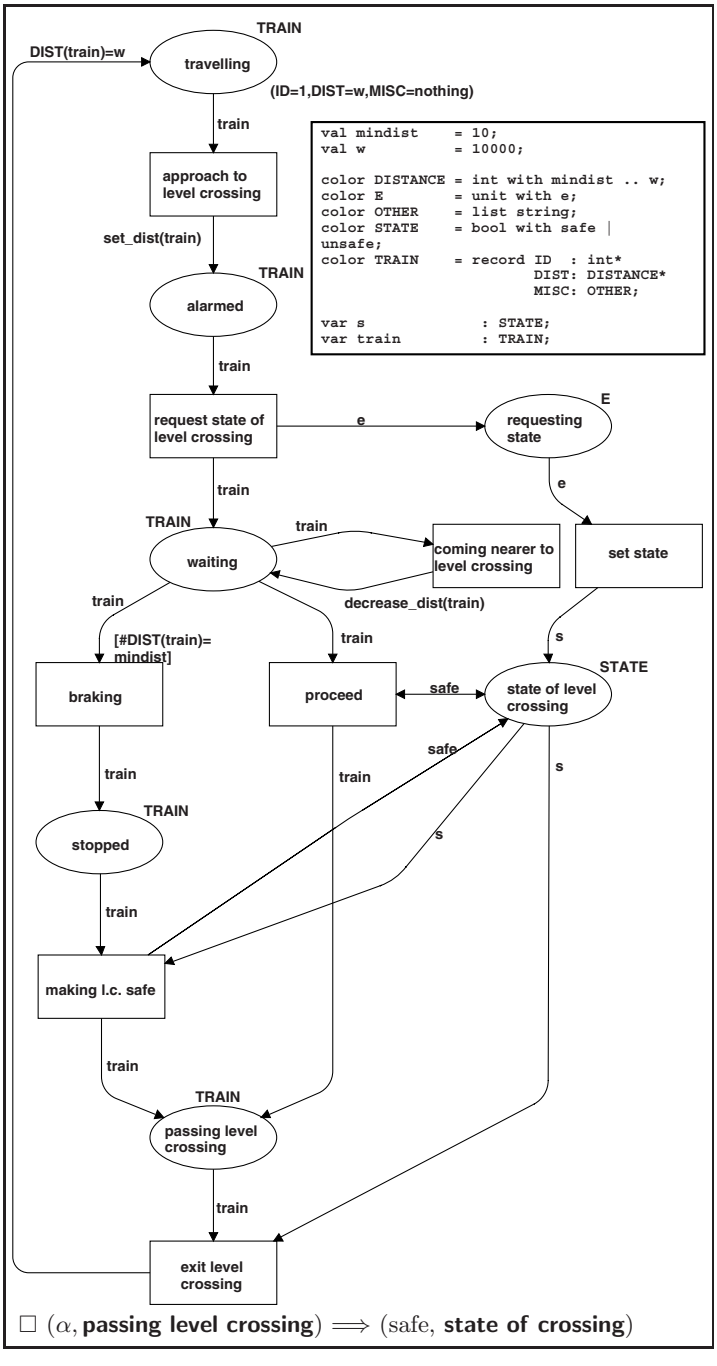


Fig. 8. CPN_1 Train passing a level crossing

Both transitions put a token coloured 'safe' back to the place **state of crossing**, such that also the conclusion is fulfilled. Next, the only transition (namely ***exit***) withdrawing that token from **state of crossing** also removes the train from the place **passing level crossing** and leads to the initial marking. Finally, as the net is cyclic and each cycle satisfies the formula (α, **passing level crossing**) $\Longrightarrow$ (safe, **state of crossing**), this formula is always satisfied, i. e. for each reachable marking. In the following we will modify, respectively enrich this model CPN_1 by complexity. Of course, we also want the resulting net to satisfy the important safety property. In other words, the already proven safety property should be preserved by the modification. Thus we employ safety property preserving and introducing transformations.

Developing Steps We are now going to enhance the model by gates for detaining road traffic from crossing the tracks. This is done by adding the coloured Petri net description of gates and connecting it to the first net CPN_1. The addition of gates to the simple model is realized by application of the rule r_{gates} depicted in the Figure 9.

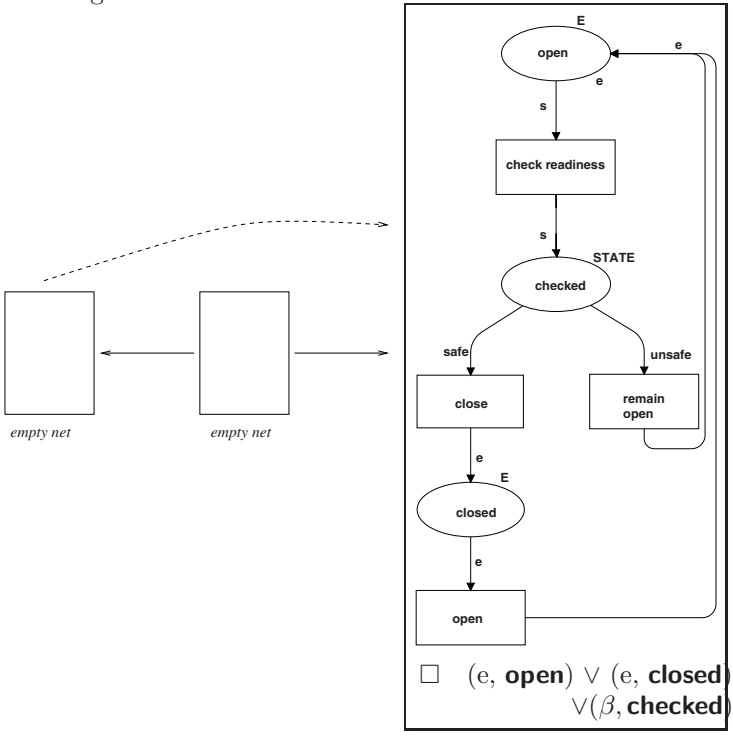


Fig. 9. Rule r_{gates} for adding gates

Initially, the gates are open. There can be a check of readiness, see transition ***check readiness***, which results in the token coloured safe, if the gates are ready, or unsafe if they are not. Depending on this result, the gates may either remain

open, or they can be closed. If so, they might be opened again leading to the initial state. Again, there is a safety property given together with the net. For $\beta \in C(\textbf{checked}) = STATE$ the safety property $\Box$ (e, **open**)$\vee$ (e, **closed**) $\vee$ $(\beta, \textbf{checked})$ informally states, that the state of gates is either open or closed or checked. Obviously, this safety property holds in the net on the right-hand side of the Figure 9. As the rule r_{gates} not only adds a subnet but also *introduces a new safety property* we call it *si-rule*, short for safety introducing. Applying this rule, of course, we do not want to lose the safety property "In any case, if a train is passing the level crossing, the level crossing is safe", which we already proved for the first net. Moreover, the introduced safety property should be propagated to the net resulting after applying the rule r_{gates}. The preservation of the old safety properties and the satisfaction of the newly introduced safety properties is stated in the next section. This means that the net gained by application of the rule r_{gates} to the first net satisfies the safety property introduced by the rule and also the originally stated safety property in the first net.

The application of the rule r_{gates} to the first net CPN_1 yields a disconnected coloured Petri net CPN_2. Intuitively, we would suppose the two transitions ***check readiness*** and ***set state*** to be somehow synchronized, i. e. the state should conform to the result of the check. Therefore, we are now going to *glue these transitions* together. This is achieved by the rule r_{glue} in the Figure 10. It is compatible with the safety properties as stated in the next section meaning that in the resulting net still all safety properties hold. The resulting net CPN_3 after applying r_{gates} to the net CPN_2.

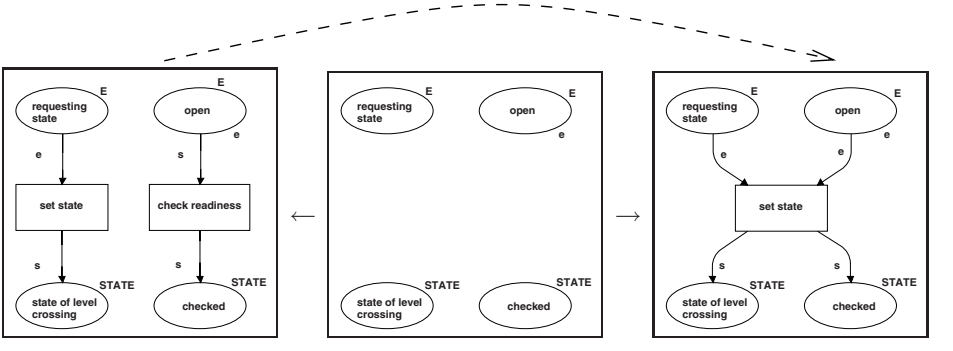


Fig. 10. Rule r_{glue} for gluing transitions

Last, we want the gates to open only after the train has left the level crossing. Therefore, we send a message to the gates indicating the exit of the train. Only then, the gates may be opened. The rule r_{msg} in the Figure 11 takes the two corresponding transitions ***exit level crossing*** and ***open*** and connects them by a place **finished**. This rule does not change the pre- and postdomains of the neighboring places and thus it is *place preserving*. By the result presented in the following section it also preserves safety properties.

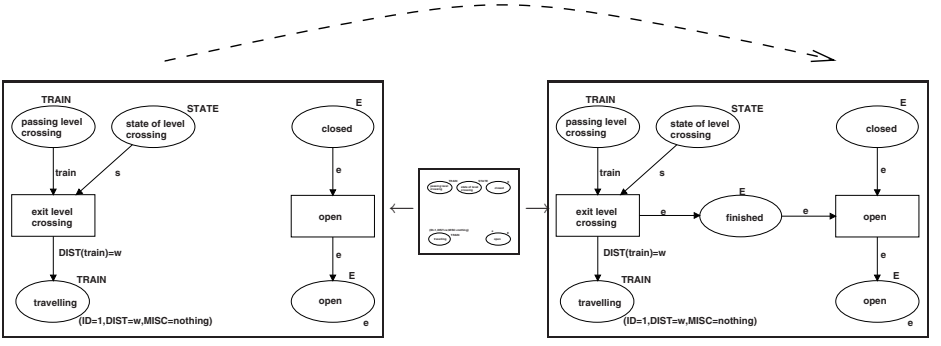


Fig. 11. Rule r_{msg}

The final net CPN_{final}, which is the result of applying the rule r_{msg} to CPN_3 is presented in Figure 12. It models a train's passage of a level crossing where the opening and closing of the gates is taken into account.

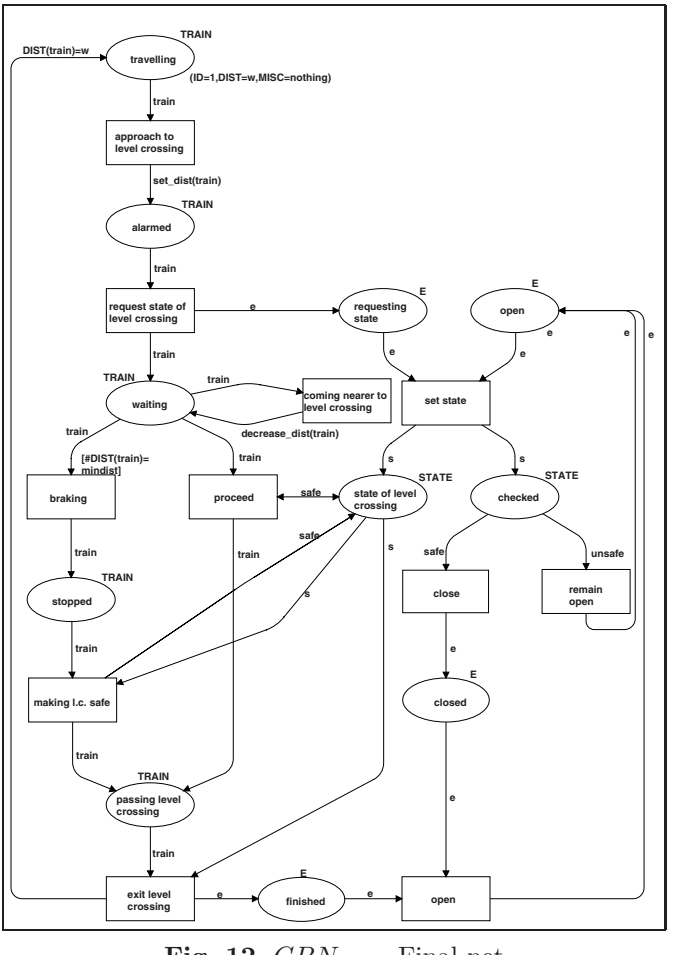
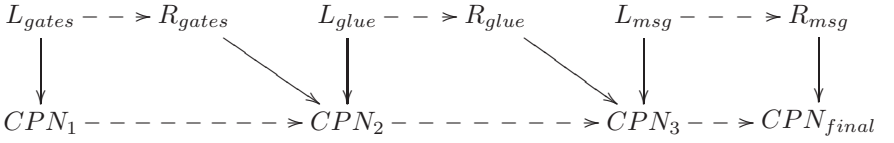


Fig. 12. CPN_{final} Final net

Summarizing, we have developed the model as follows: starting with CPN_1 in Figure 8 we iteratively applied the rules r_{gates}, r_{glue}, and r_{msg} depicted in the Figures 9, 10, and 11, respectively. A graphical representation of this transformation sequence is given below, where all arrows depict (different kinds of) morphisms:



L_r and R_r denote the left-hand side of the rule r, resp. the right-hand side of r and we omitted the interface nets for brevity. More precisely, we have gradually developed a coloured Petri net modeling a train's passage of a level crossing. We proved an important safety property that "In any case, if a train is passing the level crossing, the level crossing is safe" for the first net CPN_1. This safety property has been preserved by the subsequent refinement of the first net. Additionally, we introduced a new safety property for the gates, which then has been preserved as well. The main advantage of our approach is that we do not have to prove the safety properties in the final net but just for the first net and for the rule introducing a new safety property. By this we could add further safety properties, which were preserved by transition gluing as well as place preserving rules.

3 Morphism Classes and System Properties

In next subsections we introduce used notions on a formal basis. The achieved results concerning horizontal structuring, safety property and liveness preservation are summarized in the Section 4.

3.1 Place/Transition Nets

In this section we summarize main notions about the class of place/transition nets and its properties. The proper definitions are beyond the scope of this contribution. Presented results can be found fully elaborated in [6, 8, 17]. First we give a short intuition of the underlying basics. The precise definition can be found in [8]. Next we introduce formally different classes of net morphisms. We will also define the notions of specific systems properties such as safety properties and liveness. Finally, we list all the features which the introduced morphisms classes have. Technical details and proofs can be found in the cited papers.

Preliminaries. Here we use the algebraic notion of place/transition systems (shortly p/t systems) as introduced in [MM90]. Hence a place/transition system is given by the set of transitions and the set of places and the pre- and

post-domain function. $N = (T \underset{post}{\overset{pre}{\rightrightarrows}} P^{\oplus}, \widehat{m})$, where $P^{\oplus}$ is the free commutative monoid over P, or the set of finite multisets over P and markings are elements of $P^{\oplus}$, especially $\widehat{m}$ stands for the initial marking. The category **PTSys** of place/transition nets has plain morphisms, which consist of two functions $f = (f_T, f_P)$. Elements of the free commutative monoid over P are given mostly as finite linear sums, hence we use the extended operations $\oplus, \ominus, \leq, <>$ etc. Moreover we need to state how often a basic element is given within an element of the free commutative monoid. We define this for $p \in P$ and $w \in P^{\oplus}$, $\lambda \in \mathbb{N}$ with $w_{|p} = \lambda p \in P^{\oplus}$. This is extended to subsets $P' \subseteq P$ with $w_{|P'}$ so that there is $w = w' \oplus w''$ with $w_{|P'} = w' \in (P \setminus P')^{\oplus}$ and $w'' \in P'^{\oplus}$. The notions $\bullet e, e\bullet$ stand for sets of input and output elements of an element e (transition or place) of a net. For the firing of Petri nets we use the usual notations, that is $m[t\rangle$ means marking m enables transition t; $m[t\rangle m'$ denotes firing of transition t under marking m yielding marking m'. Reachable markings from m are given by the set $[m\rangle$. Paths of firing steps are denoted by arrows with the following special meanings $m \xrightarrow{*} m'$ arbitrary, but possibly no firing steps from m to m'. $m \xrightarrow{t_1,...,t_n} m'$ denotes firing steps using transitions $t_1, ..., t_n$. A path $m \xrightarrow{...,t,...} m'$ denotes firing steps where the transition t occurs.

Sequentially Independent Transitions [17]. Given $N = (P, T, pre, post, \widehat{m})$ a place/transition system. The set of transitions $T_I = t_1, ..., t_n \subseteq T$ is called set of *s*equentially independent transitions if the following holds:

For every reachable marking $m \in [\widehat{m}\rangle$ such that some transition $t \in T_I$ is enabled, there exists a path $m \xrightarrow{t'_1,...,t'_n} m'$ satisfying following conditions:

1. $\sum_{i=0}^{n} pre(t_i) \leq m'$
2. $m' \oplus \sum_{i=0}^{n} (post(t_i) \ominus pre(t_i)) \geq m \ominus pre(t) \oplus post(t)$
3. $t'_1, ..., t'_n \notin T_I$.

The Item 2 states that every transition sequence $t_1...t_n$ fireable from $m^1 : m[t\rangle m^1 \xrightarrow{t_1...t_n}$ is also fireable from $m' : m' \xrightarrow{t_1...t_n} m^2$.

Basic and Derived Place/Transition System-Morphisms and Categories. Given $N_i = (P_i, T_i, pre_i, post_i, \widehat{m}), i \in \{1, 2\}$ two place/transition systems.
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 [6] if the following embedding conditions hold for all $t \in T_1$ and $p \in P_1$:

1. $f_P^{\oplus}(pre_1(t)) \leq pre_2(f_T(t))$ and $f_P^{\oplus}(post_1(t)) \leq post_2(f_T(t))$
2. $f_P^{\oplus}(\widehat{m_1}_{|p}) \leq \widehat{m_2}_{|f_P(p)}$

transition preserving [6] if it is loose and the following condition holds:

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

- A transition preserving morphism is called **marking strict** [6] or **plain** [8] if $f_P^{\oplus}(\widehat{m_{1|p}}) = \widehat{m_2}_{|f_P(p)}$ for all $p \in P_1$.
- It is called **strict** if it is marking strict and injective.

place preserving [6] if it is a loose morphism and the following place preserving conditions hold:

4. $\bullet(f_P(p)) = f_T^{\oplus}(\bullet p)$ 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.
5. f_T and f_P are injective
6. $\widehat{m_2}_{|f_P} = f_P^{\oplus}(\widehat{m_1})$

transition gluing [6] if it is a loose morphism and the following holds:

7. f_P is isomorphism
8. $f_P^{\oplus}(\widehat{m_1}) = \widehat{m_2}$
9. 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.

l-transition gluing [17] if it is transition gluing and the set of transitions $T_G = \{t \in T_1 | \exists t' \neq t : f_T(t) = f_T(t')\}$ (the set of transitions not mapped bijectively) is a set of sequentially independent transitions.

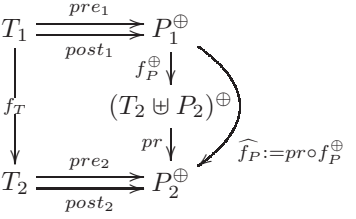
Example 1. In the Figure 3 the dashed arrow expresses a special case of a place preserving morphism. The domain of the morphism is empty, therefore all the places are preserved and new places are added.

In the Figure 6 the transition gluing morphism is used. It glues two transitions to one and keeps the rest of the net unmodified. This morphisms is also l-transition gluing as it glues transitions with empty preset and postset respectively.

In [8] 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 vicinity 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.

Abstracting Morphisms [8]. Given two place/transition systems $N_i = (T_i \rightrightarrows^{pre_i}_{post_i} P_i^{\oplus}, \widehat{m_i})$ for $i = 1, 2$. An **abstracting morphism** $f : N_1 \to N_2$ is given by $f = (f_T, f_P)$ with functions $f_T : T_1 \to T_2$ and $f_P : P_1 \to (T_2 \uplus P_2)$ such that the following conditions are satisfied:

1. for all $t \in T_1$ we have $f_P(\bullet t) = \{f_T(t)\}$
   or
   $pr \circ f_P^{\oplus} \circ pre_1(t) = pre_2(f_T(t))$
   where $pr : (T_2 \uplus P_2)^{\oplus} \to P_2^{\oplus}$ is the corresponding projection
   analogously for the *post* function.
2. for all $t_2 \in f_T(T_1)$ we have:
   $\exists t_{in} \in T_1$ 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. marking strict:
   $\forall p \in P_1$ with $f_P(p) \in P_2$ we have $f_P^{\oplus}(\widehat{m_1}_{|p}) = \widehat{m_2}_{|f_P(p)}$
4. for all $p \in P_1$ with $f_P(p) \in T_2$ we have $f_T(\bullet p) = \{f_P(p)\}$
   analogously for the *post* function.



Collapsing Subnet [8]. Given an abstracting morphism $f : N_1 \to N_2$. We have for all transitions $t \in T_2$ the collapsing subnet $subst_f(t) = (\widetilde{P}^{t,f}, \widetilde{T}^{t,f}, \widetilde{pre}^{t,f}, \widetilde{post}^{t,f}, \widehat{m_t}) \subseteq N_1$ with

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

Moreover, we define the following sets:

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

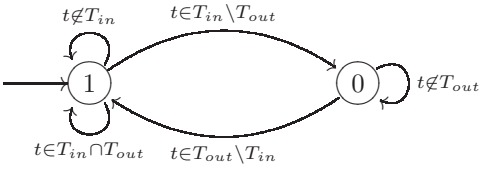
We omit the superscripts $(\_)^{t,f}$ and $(\_)^f$, if unambiguous.

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.

Live In-Out Cycle [8]. Given a place/transition system $N = (T \overset{pre}{\underset{post}{\rightrightarrows}} P^{\oplus}, \widehat{m})$. We call N *live in-out cycle* if the following conditions hold:

1. there are two distinguished subsets T_{in} and T_{out} of T, called set of in-transitions T_{in} and set of out-transitions T_{out} of N such that there is a place $c \in P$, 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 $pre(t)_{|c} = \begin{cases} 1 * c \,; t \in T_{in} \\ \epsilon \quad ; t \notin T_{in} \end{cases}$ and $post(t)_{|c} = \begin{cases} 1 * c \,; t \in T_{out} \\ \epsilon \quad ; t \notin T_{out} \end{cases}$
2. $\widehat{m}_{|c} = 1 * c$
3. N is live
4. place c is safe (1-bounded)

With respect to the number of tokens in the place c, the evolution of the live in-out cycle can be expressed in the form of a finite automaton below. The number inside the particular state is the number of tokens which reside in the place c. Due to the definition the state 1 is the initial state. From the behavior of this automaton one can infer important property of every live in-out cycle. The firing of one of the input transitions alternates (sooner or later) with the firing of one of the output transitions.



Collapsing Morphisms [8]. A **collapsing morphism** $f : N_1 \to N_2$ is an abstracting morphism which additionally satisfies the following conditions:

1. f_T is surjective,
   and f_P is quasi-surjective,i.e.
   the restriction $f_P : P_1 \setminus \mathbb{P} \to P_2$ is surjective
2. f_T and f_P are quasi-injective, i.e.
   For all $t, t' \in T_1 \setminus \mathbb{T}$ we have $f_T(t) = f_T(t')$ implies $t = t'$
   For all $p, p' \in P_1 \setminus \mathbb{P}$ we have $f_P(p) = f_P(p')$ implies $p = p'$
3. $\forall t \in \mathbb{S}$ the following holds:
   $subst_f(t)$ is a live in-out cycle with c^t the guarding place so that it is only connected to the rest of N_1 via the in- and out-transitions:
   (a) for all $t_i \in T_{in} \subseteq \widetilde{T}^t$ holds $pre_2(t) = \widehat{f_P}(pre_1(t_i) \ominus c^t)$
   (b) for all $t_o \in T_{out} \subseteq \widetilde{T}^t$ holds $post_2(t) = \widehat{f_P}(post_1(t_o) \ominus c^t)$
   (c) for all $t_s \in \widetilde{T}^t \setminus T_{in}$ holds $\bullet t_s \subseteq \widetilde{P}^t$
   (d) for all $t_s \in \widetilde{T}^t \setminus T_{out}$ holds $t_s \bullet \subseteq \widetilde{P}^t$

Example 2. In the Figure 4 the dashed arrow from right-hand side to the left-hand side of the depicted rule is a collapsing morphisms as it collapses a subnet to a one transition. This morphism is also abstracting.

Next, we specify the meaning of a safety property and liveness in order to be able to present our main results concerning preservation (translation) of these properties. We recall safety formulas over markings (in the sense of [MP92]), their translations via morphisms and used notion of liveness.

Safety Properties, Translations [6]. Consider a place/transition system N with a set of places P.

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

$$\begin{aligned} \varphi_1 \in \mathcal{F} &\Longrightarrow \neg\varphi_1 \in \mathcal{F}, \\ \varphi_1 \in \mathcal{F}, \varphi_2 \in \mathcal{F} &\Longrightarrow \varphi_1 \wedge \varphi_2 \in \mathcal{F} \end{aligned}$$

   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:

$$\begin{aligned} m \models_N \lambda p &\text{ iff } \lambda p \leq m, \\ m \models_N \neg\varphi_1 &\text{ iff } \neg(m \models_N \varphi_1) \text{ and} \\ m \models_N \varphi_1 \wedge \varphi_2 &\text{ iff } (m \models_N \varphi_1) \wedge (m \models_N \varphi_2). \end{aligned}$$

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 states reachable from m:

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

   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 morphism $f = (f_P, f_T) : N_1 \to N_2$ 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

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

4. The *l-translation* (in [17]) $\mathcal{LT}_f$ of formulas over N_2 along a collapsing morphism $f = (f_T, f_P) : N_1 \to N_2$ to formulas over N_1 is given for atoms by

$$\begin{aligned} \mathcal{LT}_f(\lambda p) &= \lambda f_P^{-1}(p) && if (p\bullet) \cap \mathbb{S}^f = \emptyset, \\ \mathcal{LT}_f(\lambda p) &= \lambda f_P^{-1}(p) \vee \bigvee_{t_i \in (p\bullet) \cap \mathbb{S}^f} \neg(1 * c^{t_i}), && if (p\bullet) \cap \mathbb{S}^f \neq \emptyset. \end{aligned}$$

   The l-translation of formulas is given recursively by

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

Liveness. A place/transition system $N = (T, P, pre, post, \widehat{m})$ is called *live* if for arbitrary $m_1 \in [\widehat{m_1}\rangle$ and arbitrary $t_1 \in T_1$ there exists some $m_1' \in [m_1\rangle$ such that $m_1'[t_1\rangle$.

Theorem *Properties of Morphisms.*

1. *Place Preserving Morphisms Preserve Safety Properties [6]*
   Let $f : N_1 \to N_2$ be a place preserving morphism . Let $\Box\varphi$ be a safety property. Then the following holds:
   $$N_1 \models \Box\varphi \Longrightarrow N_2 \models \mathcal{T}_f(\Box\varphi)$$
2. *Transition Gluing Morphisms Preserve Safety Properties [6]*
   Let $f : N_1 \to N_2$ be a transition gluing morphism and let $\Box\varphi$ be a safety property then we have
   $$N_1 \models \Box\varphi \Longrightarrow N_2 \models \Box\varphi$$
3. *L-Transition Gluing Morphisms Preserve Liveness [17]*
   Given $f : N_1 \longrightarrow N_2$ a l-transition gluing morphism and let N_1 be live. Then the net N_2 is live as well.
4. *Collapsing Morphisms Respect Liveness [8]*
   Given a collapsing morphisms $f : N_1 \to N_2$ and let N_2 be live. Then the net N_1 is live as well.
5. *Collapsing Morphisms Translate (Backwards) Safety Properties [17]*
   Let $f : N_1 \to N_2$ be a collapsing morphism. Let $\Box\varphi$ be a safety property. Then the following holds:
   $$N_2 \models \Box\varphi \Longrightarrow N_1 \models \mathcal{LT}_f(\Box\varphi)$$

Achieved results based on foregoing theorems are summarized in the Section 4.

3.2 Coloured Petri Nets

In this section we are going to present the results leading to preservation of safety properties in coloured Petri nets by transformations. We use coloured Petri nets [Jen92] and define different kinds of morphisms. These morphisms have different properties and are used for different purposes. We define formulas expressing safety properties and their preservation by a special kind of morphism, the safety preserving morphisms.

Coloured Petri nets [Jen92]. A **non-hierarchical coloured Petri net** is a tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ according to the description below:

1. Σ is a finite set of non-empty types, called **colour sets**.
2. P is a finite set of **places**.
3. T is a finite set of **transitions**.

4. A is a finite set of **arcs** s.t:
   $P \cap T = P \cap A = T \cap A = \emptyset$.
5. $N : A \to P \times T \cup T \times P$ is a **node** function.
6. $C : P \to \Sigma$ is a **colour** function.
7. $G : T \to \mathbb{EXP}_{\mathbb{B}}(\Sigma)$ is a **guard** function, where $\mathbb{EXP}_{\mathbb{B}}(\Sigma)$ denotes Boolean expressions over the colour sets in Σ.
8. $E : A \to \mathbb{EXP}(\Sigma)_{MS}$ is an **arc expression** function, where $\mathbb{EXP}(\Sigma)$ denotes expressions over the colour sets in Σ and $(\_)_{MS}$ multi-sets thereover.
9. $I : P \to \mathbb{EXP}_{\text{closed}}(\Sigma)_{MS}$ is an **initialization** function, where $\mathbb{EXP}(\Sigma)$ denotes closed expressions over the colour sets in Σ and $(\_)_{MS}$ multi-sets thereover.

A marking of a coloured Petri net is given as a multiset over the set of token elements $TE = \{(c, p) | p \in P \text{ and } c \in C(p)\}$.

Safety Property Preserving Morphisms

The morphisms we now introduce are based on mappings of the net components. That is, a CPN-morphism maps places to places, transitions to transitions, arcs to arcs, and expressions over colour sets to expressions over colour sets. Moreover, we distinguish different kinds of morphisms that have different properties.

CPN-Morphisms. A CPN-morphism $f : CPN_1 \to CPN_2$ with coloured Petri nets $CPN_i = (\Sigma_i, P_i, T_i, A_i, N_i, C_i, G_i, E_i, I_i)$ for $i = 1, 2$ is given by the mappings of the components $f = (f_{\Sigma,} f_P, f_T, f_A)$ with
$f_{\Sigma,} \mathbb{EXP}(\Sigma_1) \to \mathbb{EXP}(\Sigma_2)$, $f_P : P_1 \to P_2$, $f_T : T_1 \to T_2$, and $f_A : A_1 \to A_2$.
A CPN-morphism is called:

loose if the following *compatibility conditions* hold:

1. $N_2 \circ f_A = f_{PT} \circ N_1$
   where $f_{PT} : P_1 \times T_1 \cup T_1 \times P_1 \to P_2 \times T_2 \cup T_2 \times P_2$
   with $f_{PT}(x, y) := \begin{cases} (f_P(p), f_T(t)) & (x, y) = (p, t) \\ (f_T(t), f_P(p)) & (x, y) = (t, p) \end{cases}$
   That means, that the following diagram commutes:

$$\begin{array}{ccc} A_1 & \xrightarrow{N_1} & P_1 \times T_1 \cup T_1 \times P_1 \\ {\scriptstyle f_A}\downarrow & & \downarrow{\scriptstyle f_{PT}} \\ A_2 & \xrightarrow{N_2} & P_2 \times T_2 \cup T_2 \times P_2 \end{array}$$

2. $C_2 \circ f_P = f_{\Sigma_\circ} C_1$
3. $G_2 \circ f_T = f_{\Sigma_\circ} G_1$
4. $E_2 \circ f_A = f_{\Sigma_\circ} E_1$
5. $f_{\Sigma(} I_1(p)) \leq I_2(f_P(p))$

transition preserving if it is loose and the *transition preserving condition* holds:

6. no "new" arc is connected to "old" transitions, that is
   for all $A \in A_2 \backslash f_A(A_1)$ we have :

$$N_2(A) \in P_2 \times T_2 \backslash f_T(T_1) \ \cup \ T_2 \backslash f_T(T_1) \times P_2$$

A special case are strict morphisms with $f_{\Sigma(}I_1(p)) = I_2(f_P(p))$

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

7. no "new" arc is connected to "old" places, that is
   for all $A \in A_2 \backslash f_A(A_1)$ we have :

$$N_2(A) \in P_2 \backslash f_P(P_1) \times T_2 \ \cup \ T_2 \times P_2 \backslash f_P(P_1)$$

8. $\sum_{p \in P_1} f_{\Sigma(}I_1(p)) \leq \sum_{p \in P_2} I_2(p)$
9. f_P, f_T are injective
10. f_Σ is persistent in the sense of [EM85], meaning intuitively, that Σ_2 is a consistent enrichment of Σ_1

transition gluing if it is loose and the *transition gluing conditions* hold:

11. f_A is bijective
12. f_T is surjective
13. f_P and f_Σ are identities
14. $I_1 = I_2$

Remark 1. Conditions 1 - 4 ensure the compatibility of componentwise mappings. Condition 5 states that the initialization function of the target net is place-wise greater than the source net. The difference to condition 8 concerns the identification of places by non-injective f_P. Condition 6 guarantees that the neighborhood of transitions remains unchanged, whereas condition 7 ensures this for places. Condition 11 ensures that the identification of transitions by non-injective f_T does not affect the adjacent arcs.

Example 3. In the Figures 9, 10 and 11 the morphisms denoted by $\longrightarrow$ are transition preserving, as there are no transitions in the interfaces of the rules. The morphism denoted by $- - \succ$ is place preserving, as in the right-hand side of the corresponding rule all new arcs are connected only to new places. The morphism denoted by $\cdots\cdots\succ$ is a transition gluing morphism.

Safety Properties. We use a simple kind of temporal logic formulas over the marking of the places in order to express safety properties. We merely employ the always operator $\Box$ as the outermost operator. This yields safety properties in the sense of [MP92]. More precisely, we have: $\lambda(c, p)$ is a static formula for $\lambda \in \mathbb{N}$, $p \in P$, and $c \in C(p)$. Static formulas are generated using the logical operators $\wedge$ and $\neg$. $\Box\varphi$ is a safety property, where φ is a static formula.
The validity of formulas is given w. r. t. the marking. Let M be a multi-set over the set of token elements $TE = \{(c, p) | p \in P \text{ and } c \in C(p)\}$, e. g. an arbitrary marking m of a coloured Petri net CPN then

$$CPN \models_m \lambda(c,p) \text{ if and only if } \lambda(c,p) \leq m.$$

For $CPN \models_m \neg\varphi_1$ if and only if $\neg(CPN \models_m \varphi_1)$ and $CPN \models_m (\varphi_1 \wedge \varphi_2)$ if and only if $CPN \models_m \varphi_1 \wedge CPN \models_m \varphi_2$. The safety property $\Box\varphi$ holds in CPN under m if and only if φ holds in all states reachable from m:

$$CPN \models_m \Box\varphi \text{ if and only if } \forall m' \in [M\rangle : CPN \models_{m'} \varphi.$$

The translation of formulas $\mathcal{T}_f$ over CPN_1 to formulas over CPN_2 along a morphism $f : CPN_1 \to CPN_2$ is given for atoms by

$$\mathcal{T}_f(\lambda(c,p)) = \lambda(f_\Sigma(c), f_P(p)).$$

The translation of formulas is given recursively by $\mathcal{T}_f(\neg\varphi) = \neg\mathcal{T}_f(\varphi)$, and $\mathcal{T}_f(\varphi_1 \wedge \varphi_2) = \mathcal{T}_f(\varphi_1) \wedge \mathcal{T}_f(\varphi_2)$, and $\mathcal{T}_f(\Box\varphi) = \Box\mathcal{T}_f(\varphi)$

Theorem *Properties of Morphisms.*

1. *Place Preserving Morphisms Preserve Safety Properties [12].*
   A place preserving morphism $f : CPN_1 \to CPN_2$ preserves safety properties, i.e. for all safety properties $\Box\varphi$ we have :

$$CPN_1 \models \Box\varphi \Longrightarrow CPN_2 \models \mathcal{T}_f(\Box\varphi)$$

2. *Transition Gluing Morphisms Preserve Safety Properties [12].*
   A transition gluing morphism $f : CPN_1 \to CPN_2$ preserves safety properties, i.e. for all safety properties $\Box\varphi$ we have :

$$CPN_1 \models \Box\varphi \Longrightarrow CPN_2 \models \mathcal{T}_f(\Box\varphi)$$

4 Summary of Results

In this section we summarize the results we have obtained for rule-based refinement of Petri nets. Moreover, we give for these results an interpretation of its meaning. For proofs and technical details we refer to the corresponding papers.

The presented results comprise preservation of system properties via transformations by different rules and compatibility of transformations with horizontal structuring.

4.1 Preservation of Properties

Subsequently we refer to conference papers [15, 11, 16], but due to space limitations there many proofs are given in more detail in the corresponding technical reports [6, 12, 8].

Theorem *[15, 11].* Given a rule $(r = (L \leftarrow K \rightarrow R), f : L \to R)$ with f being

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

then for a transformation step $N_1 \overset{(r,f)}{\Longrightarrow} N_2$ holds:

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

These rules are called *sp-rules.*

Theorem *[15, 11].* Given a rule (r, f) with $r = (\emptyset \leftarrow \emptyset \rightarrow R)$ so that $R \models \Box\varphi$ with a occurrence $g : R \rightarrow N_2$. Then for a transformation step $N_1 \overset{(r,f)}{\Longrightarrow} N_2$ holds:

$$N_2 \models \mathcal{T}_g(\Box\varphi).$$

We call these rules *si-rules.*

Theorem *[16, 17].* Given a rule $(r = (L \leftarrow K \rightarrow R), f : R \rightarrow L)$ with f being a collapsing morphism. Then for a transformation step $N_1 \overset{(r,f)}{\Longrightarrow} N_2$ holds:

$$N_1 \models \Box\varphi \Longrightarrow N_2 \models \mathcal{LT}_f(\Box\varphi)$$

$$N_1 \quad live \quad \Longrightarrow \quad N_2 \quad live^1.$$

We call these rules *lp-rules.*

Theorem *[17].* Given a rule $(r = (L \leftarrow K \rightarrow R), f : L \rightarrow R)$ with f being a l-transition gluing morphism. Then for a transformation step $N_1 \overset{(r,f)}{\Longrightarrow} N_2$ holds:

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

These rules are also called *lp-rules.*

Theorem *[17].* Given rule (r, f) with $r = (\emptyset \leftarrow \emptyset \rightarrow R)$ so that R is live. Then for a transformation step $N_1 \overset{(r,f)}{\Longrightarrow} N_2$ holds:

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

These rules are called *li-rules.*

Interpretation. Foregoing paragraphs contain an overview of achieved results. The interpretation of these theorems can be done in a succinct way as proof-rules, which are of interest for model developers.

Consider a net transformation $N_1 \overset{(r,f)}{\Longrightarrow} N_2$. Then there are the following proof-rules (according to the results):

$$\frac{(r, f) \text{ is a sp-rule; } N_1 \text{ satisfies } \Box\varphi}{N_2 \text{ satisfies } \Box\mathcal{T}_f(\varphi)}$$

$$\frac{(r, f : L \rightarrow R) \text{ is a si-rule; } g : R \rightarrow N_2; \ R \text{ satisfies } \Box\varphi}{N_2 \text{ satisfies } \mathcal{T}_g(\Box\varphi)}$$

[1] under appropriate occurrence morphisms, see [8] for details

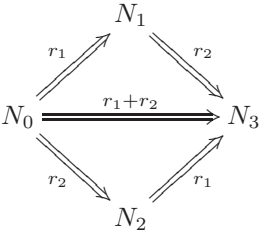
$$\frac{(r, f) \text{ is a lp-rule; } N_1 \text{ is live}}{N_2 \text{ is live}^1}$$

$$\frac{(r, f : L \to R) \text{ is a li-rule; } R, N_1 \text{ are live}}{N_2 \text{ is live}}$$

$$\frac{(r, f) \text{ is a lp-rule; } N_1 \text{ satisfies } \Box\varphi}{N_2 \text{ satisfies } \Box\mathcal{LT}_f(\varphi) \text{ resp. } \Box\mathcal{T}_f(\varphi)}$$

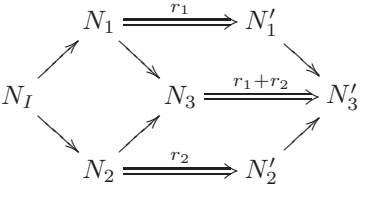
4.2 Compatibility Results

Theorem *Church-Rosser and Parallelism.* Given two rules r_1 and r_2 that satisfy certain independence conditions (see [2]) then we have: If we can apply r_1 and r_2 sequentially to N_0 and then to N_1, then we can apply them in parallel as well as in the opposite order.

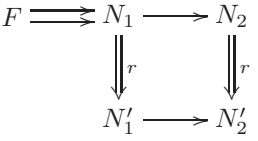


Interpretation. These Church-Rosser theorems state a local confluence in the sense of formal languages. In the context of specification techniques they are important as they state conditions for the independent development of different parts or views of the system. These independence conditions hold for refinement rules as well. The parallel independence states that the matches of both rules overlap in parts that are not deleted only. Sequential independence states that those parts created by the first transformation step are not deleted in the second.

Theorem *Union and Fusion Theorems.* Union theorem [9] states that - provided that certain independence conditions hold - given two rules r_1 and r_2 and a union $N_1 \leftarrow N_I \rightarrow N_2$ of N_1 and N_2 then we obtain the same result (up to isomorphism) whether we derive first $N_1 \overset{r_1}{\Longrightarrow} N_1'$ and $N_2 \overset{r_2}{\Longrightarrow} N_2'$ and then construct the union of N_1' and N_2' resulting in N_3' or whether we first construct the union N_3 of N_1 and N_2 followed by parallel transformation step $N_3 \overset{r_1+r_2}{\Longrightarrow} N_3'$.



Fusion theorem [9] is expressed similarly. Provided that certain independence conditions hold and given a rule r and a fusion $F \rightrightarrows N_1$ then we obtain the same result (up to isomorphism) 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 $F \rightrightarrows N_1$ resulting in N_2 and then perform the transformation step $N_2 \overset{r}{\Longrightarrow} N_2'$.



Interpretation. Both theorems state that transformation of Petri nets is compatible with the corresponding structuring technique under reasonable independence conditions. These conditions ensure that the gluing affects those subnets only that remain unchanged.

4.3 Open Problems

We have introduced several morphisms that preserve some property of the Petri net. So we obtained various morphisms classes and different categories. Based on these morphisms we can define rules and transformations that preserve stated property as well. The open question is how do the morphisms interact?

- Different morphisms can be used for the same development as exemplified in our examples in Section 2. We can use different proof-rules for the deduction of certain property although we achieve the transformations in different categories.
- Another question is whether it is possible to find adequate morphism classes that comprise the morphisms we have not yet defined and at the same time still preserve those properties. We do not think so, as morphisms would need to be so general that they will not preserve anything.
- Last but not least we think it would be promising to extend the abstract theory in order to cope with different morphism classes.

Very often are properties of communicating and reactive systems stated as temporal logic formulas. Although safety-properties are such formulas, they do not provide necessary expressive power. The open problem is how to integrate temporal formulas preservation into the rule-based refinement.

5 Conclusion

We conclude with a summary and a discussion of future work. Summarizing, we have for different net classes the following results for rule-based refinement in table 1. This table illustrates we have already developed quite a complex theory. There are various possibilities how to proceed open. Future work can comprise among others the following activities.

Table 1. Achieved results

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	√	?	?
Independence Conditions	√	√	√
Church Rosser I + II	√	√	√
Parallelism	√	√	√
Union	√	√	√
Fusion	√	√	√

- The extension of our approach of liveness preserving transformation to high-level Petri nets is an obvious way to continue. The basic ideas remain the same but the data type part has to be dealt with.
- Other system properties, especially liveness properties in the sense of temporal logic [MP92, Lam94] are most promising to integrate into this approach. In order to do so, morphisms have to be defined that preserve such properties. Then a specific set of conditions have to be satisfied in order to achieve similar results for rules and transformations.
- Tool support is for the practical use the main precondition. The involved morphisms are quite complex, so the user needs tool support for defining and applying rules. The tool should assist the choice as well as the execution of rules and transformations.
- The integration of the various morphisms classes as discussed in Subsection 4.3 is one of the demanding and urgent tasks.
- Collapsing morphisms are very specialized and thus quite restrictive. We think that it is possible to extend the notion of collapsing morphisms in a natural way in order to obtain more expressive notion of liveness respecting morphisms.
- Preservation of system properties under horizontal structuring is also of interest for future investigation.

References

[BDH92] E. Best, R. Devillers, and J. Hall. The Box Calculus: a new causal algebra with multi-label communication. In *Advances in Petri Nets*, pages 21–69. Lecture Notes in Computer Science 609, 1992. 167

[BGV91] W. Brauer, R. Gold, and W. Vogler. A Survey of Behaviour and Equivalence Preserving Refinements of Petri Nets. *Advances in Petri Nets*, Lecture Notes in Computer Science 483:1–46, 1991. 167, 168

[BS90] J. Bradfield and C. Stirling. Verifying temporal properties of processes. In J. C. M. Baeten et al., editor, *Lecture Notes in Computer Science; CONCUR'90, Theories of Concurrency: Unification and Extension. (Conference, 1990, Amsterdam, The Netherlands)*, pages 115–125, Berlin, Germany, 1990. Springer Verlag. 167

[CT90] Y. Chen and W. T. Tsai. An algebraic approach to Petri net reduction and its application to protocol analysis. Technical report, University of Minnesota, 1990. 167

[DA92] R. David and H. Alla, editors. *Petri Nets and Grafcet*. Prentice Hall (UK), 1992. 167

[DDGJ90] W. Damm, G. Döhmen, V. Gerstner, and B. Josko. Modular verification of petri nets: The temporal logic approach. In J. W. de Bakker et al., editors, *Lecture Notes in Computer Science; Proceedings of the REX Workshop on Stepwise Refinement, 1989, Mook, The Netherlands*, pages 180–207, Berlin, Germany, 1990. Springer-Verlag. 167

[DM90] J. Desel and A. Merceron. Vicinity Respecting Net Morphisms. In *Advances in Petri Nets*, pages 165–185. Springer Verlag, 1990. Lecture Notes in Computer Science 483. 167, 180

[Ehr97] H. Ehrig et al. Integration von Techniken der Software Spezifikation für ingenieuwissenschaftliche Anwendungen. Antrag für ein Schwerpunkprogramm an die DFG, http://tfs.cs.tu-berlin.de/SPP/index.html, 1997. (akzeptiert als DFG-SPP von Januar 1998 bis Dezember 2003). 172

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, Berlin, 1985. 186

[ERW02] H. Ehrig, W. Reisig, and H. Weber et al. The Petri Net Baukasten of the DFG-Forschergruppe PETRI NET TECHNOLOGY. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors. *Advances in Petri Nets: Petri Net Technologies for Modeling Communication Based Systems*. LNCS. Springer, 2002. To Appear. 162, 163

[ES91] J. Esparza and M. Silva. On the analysis and synthesis of free choice systems. *Lecture Notes in Computer Science; Advances in Petri Nets 1990*, 483:243–286, 1991. 167, 168

[Esp94] J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, 23:151–195, 1994. 167, 168

[FWL] J. Favrel, H. Wu, and K. H. Lee. Reduction method of coloured Petri nets. In *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*. 167

[Gen91] H. J. Genrich. Predicate/Transition Nets. In *High-Level Petri Nets: Theory and Application*, pages 3–43. Springer Verlag, 1991. 172

[GG90] R. J. van Glabbeck and U. Golz. Equivalences and Refinement. In *Semantics of Systems of Concurrent Processes*, pages 309–333. Springer Verlag, 1990. Lecture Notes in Computer Science 469. 167

[GL81] H. J. Genrich and K. Lautenbach. System Modelling with High-Level Petri Nets. *Theoretical Computer Science*, 13:109–136, 1981. 172

[HRH91] R. R. Howell, L. E. Rosier, and Chun Yen Hsu. A taxonomy of fairness and temporal logic problems for Petri nets. *Theoretical Computer Science*, 82(2):341–372, 1991. 167

[Jan97] L. Jansen. Referenzfallstudie Verkehrsleittechnik. http://www.ifra.ing.tu-bs.de/ m33/spezi/, 1997. 172

[JCHH91] K. Jensen, S. Christensen, P. Huber, and M. Holla. *Design/CPN. A Reference Manual*. Meta Software Cooperation, 125 Cambridge Park Drive, Cambridge Ma 02140, USA, 1991. 172

[Jen92] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1: Basic Concepts. Springer Verlag, EATCS Monographs in Theoretical Computer Science edition, 1992. 172, 184

[Jen94] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, volume 2: Analysis Methods. Springer Verlag, EATCS Monographs in Theoretical Computer Science edition, 1994. 167, 172

[Jen97] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, volume 3: Practical Use. Springer Verlag, EATCS Monographs in Theoretical Computer Science edition, 1997. 172

[Lam94] L. Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems 16*, 3:872–923, 1994. 191

[MM90] J. Meseguer and U. Montanari. Petri Nets are Monoids. *Information and Computation*, 88(2):105–155, 1990. 178

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems, Specification*. Springer Verlag, 1992. 167, 183, 186, 191

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

[Peu02] S. Peuker. Concurrency based transition refinement for the verification of distributed algorithms. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors. *Advances in Petri Nets: Petri Net Technologies for Modeling Communication Based Systems*. LNCS. Springer, 2002. To Appear. 162

[Rei91] W. Reisig. Petri Nets and Algebraic Specifications. *Theoretical Computer Science*, 80:1–34, 1991. 172

[Sch96] K. Schmidt. *Symbolische Analysemethoden für algebraische Petri-Netze*, volume 4. Bertz Verlag, versal edition, 1996. 167

[Sou91] Y. Soussy. Deterministic systems of sequential processes: a class of structured Petri nets. In *Applications and Theory of Petri Nets, Gjern, Juni 1991*, pages 62–81. Springer Verlag, 1991. 168

[Vau87] J. Vautherin. Parallel System Specification with Coloured Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets 87*, pages 293–308. Springer Verlag, 1987. Lecture Notes in Computer Science 266. 172

[vdA97] W. M. P. van der Aalst. Verification of workflow nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets*, volume 1248 of *LNCS*, pages 407–426. Springer-Verlag, June 1997. 168

[vdA98] W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8:21–66, 1998. 168

[vdAtH98] W. M. P. van der Aalst and A. H. M. ter Hofstede. Verification of Workflow Task Structures: A Petri-net-based approach. Forschungsberichte des AIFB 380, Universität Karlsruhe, November 1998. 168

[WER99] H. Weber, H. Ehrig, and W. Reisig, editors. *Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems, Part II: The "Petri Net Baukasten"*. Fraunhofer Gesellschaft ISST, October 1999. 162

[WER01] H. Weber, H. Ehrig, and W. Reisig, editors. *2nd Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems*, Berlin, Germany, Sept. 2001. Researcher Group Petri Net Technology, Fraunhofer Gesellschaft ISST. 162

[WLB01] H. Weber, S. Lembke, and A. Borusan. Improving the Usability of Petri Nets with the Petri Net Baukasten. In H. Ehrig, G. Juhás, J. Padberg, G. Rozenberg, editors, *Advances in Petri Nets: Unifying Petri Nets*, volume 2128 of *LNCS*, pages 54–78. Springer-Verlag, 2001.

Our Work on Transformations and Rule-Based Refinement

[1] H. Ehrig, M. Gajewsky, and F. Parisi-Presicce. *High-Level Replacement Systems with Applications to Algebraic Specifications and Petri Nets*, chapter 6, pages 341–400. Number 3: Concurrency, Parallelism, and Distribution in Handbook of Graph Grammars and Computing by Graph Transformations. World Scientific, 1999. 163

[2] 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, LNCS 532*, pages 269–291. Springer Verlag, 1991. 168, 189

[3] 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. 163

[4] C. Ermel, J. Padberg, and H. Ehrig. Requirements engineering of a medical information system using rule-based refinement of petri nets. In *Proc. IDPT Conference (International Design and Process Technology)*, pages 186–193, 1996. 171

[5] M. Gajewsky. Concepts and Requirements for Transformations within Petri Net Based Process Models. In A. Ertas, editor, 5^{th} *World Conference on Integrated Design and Process Technology, Special Session on Model Integration*, 2000. CD-ROM, 8 pages. 163

[6] M. Gajewsky, K. Hoffmann, and J. Padberg. Place Preserving and Transition Gluing Morphisms in Rule-Based Refinement of Place/Transition Systems. Technical Report 99-14, Technical University Berlin, 1999. 178, 179, 180, 183, 184, 187

[7] M. Gajewsky and F. Parisi-Presicce. On Compatibilty of Model and Class Transformations . In M. Cerioli, and G. Reggio, editors, 15^{th} *International Workshop on Algebraic Development Techniques and General Workshop of the CoFI WG*, Lecture Notes in Computer Science 2267. Springer Verlag, 2001. 163

[8] 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. 167, 178, 180, 181, 182, 184, 187, 188

[9] J. Padberg. Categorical Approach to Horizontal Structuring and Refinement of High-Level Replacement Systems. *Applied Categorical Structures*, 7(4):371–403, December 1999. 163, 165, 171, 189, 190

[10] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic high-level net transformation systems. *Mathematical Structures in Computer Science*, 5:217–256, 1995. 172

[11] J. Padberg and M. Gajewsky. Rule-Based Refinement of Petri Nets For Modeling Train Control Systems. In Š. Kozák and M. Huba, editors, *Petri Nets in Design, Modelling and Simulation of Control Systems, Special Session at the IFAC Conference on Control Systems Design*, pages 299–304, 2000. 168, 187, 188

[12] J. Padberg and M. Gajewsky. Safety Preserving Transformations of Coloured Petri Nets. Technical Report 2000-13, Technical University Berlin, 2000. 187

[13] J. Padberg, M. Gajewsky, and C. Ermel. Rule-Based Refinement of High-Level Nets Preserving Safety Properties. In E. Astesiano, editor, *Fundamental Approaches to Software Engineering*, pages 221–238. Springer Verlag, Lecture Notes in Computer Science 1382, 1998. 163, 167

[14] J. Padberg, M. Gajewsky, and C. Ermel. Rule-based refinement of high-level nets preserving safety properties. *Science of Computer Programming*, 40:97–118, 2001. www.elsevier.nl/locate/scico. 167

[15] J. Padberg, K. Hoffmann, and M. Gajewsky. Stepwise Introduction and Preservation of Safety Properties in Algebraic High-Level Net Systems. In T. Maibaum, editor, *Fundamental Approaches to Software Engineering*, pages 249–265. Springer Verlag, Lecture Notes in Computer Science 1783, 2000. 167, 187, 188

[16] M. Urbášek and J. Padberg. Preserving liveness with rule-based refinement of place/transition systems. In Society for Design and Process Science (SDPS), editors, *Proc. IDPT 2002: Sixth World Conference on Integrated Design and Process Technology*, 2002. to appear. 167, 187, 188

[17] M. Urbášek. Another Safety Property and Liveness Preserving Morphisms of P/T Systems. Technical Report, Technical University of Berlin, 2002. 178, 179, 180, 183, 184, 188

A Notation

Basics	
$\mathbb{N}$	natural numbers
$\mathcal{P}$	power set
Free commutative monoids	
$P^{\oplus}$	free commutaive monoid over the set P
$\oplus, \ominus$	operations on monoid elements
$\leq, <>$	comparison predicates on monoid elements
$w_{\|p}, w_{\|P'}$	restrictions of monoid elements with $w_{\|P'} := w'$ for $w \in P^{\oplus}$ and $w = w' \oplus w''$ where $w' \in P'^{\oplus}$ and $w'' \in (P \backslash P')^{\oplus}$
Petri nets	
$\widehat{m}$	initial marking
$m \xrightarrow{*} m'$	firing path from m to m'
$m[t > m'$	firing of t from m to m'
$m[t >$	t is enabled under m
$m' \in [m >$	m' is reachable from m
$\bullet x, x\bullet$	pre- and post-set of a place or a transition x
c, c^t	guarding place
Petri net examples	
$\mathbf{p}$	place $\mathbf{p}$
$\boldsymbol{t}$	transition $\boldsymbol{t}$
Morphisms	
$f : N_1 \longrightarrow\!\!\!\!\!\rightarrow N_2$	transition preserving Petri net morphism (i.e. mapping transition onto transition, place onto place)
$f : N_1 \dashrightarrow\!\!\!\!\!\rightarrow N_2$	property preserving or property respecting morphism
System Properties	
λp	atomic formula over a place/transition net with $\lambda \in \mathbb{N}$ and $p \in P$
$\lambda(c, p)$	atomic formula over a coloured Petri net with $\lambda \in \mathbb{N}$, $p \in P$, and $c \in C(p)$
φ	static formula generated over atomic ones using the logical operators $\wedge$ and $\neg$
$\Box\varphi$	safety formula
$N \models_{\widehat{m}} \Box\varphi$	net N with the initial marking $\widehat{m}$ satisfies $\Box\varphi$
$\mathcal{T}_f$	translation of formulas over a net N_1 along a morphism $f : N_1 \to N_2$
$\mathcal{LT}_f$	l-translation of formulas over a net N_2 along a collapsing morphism $f : N_1 \to N_2$
High-Level Replacement Sytems	
$r = (L \leftarrow K \rightarrow R)$	rule
$N_1 \overset{r}{\Longrightarrow} N_2$	transformation step using rule r
$N_1 \overset{(r,f)}{\Longrightarrow} N_2$	property preserving transformation step using rule r and property preserving morphism f
$N_1 \overset{*}{\Longrightarrow} N_2$	transformation sequence