Introduction to Universal Parameterized Net Classes[*]

Hartmut Ehrig   Julia Padberg

Technische Universität Berlin,
Sekretariat FR 6-1, Franklinstr. 28/29, D-10587 Berlin,
email: {ehrig,padberg}@cs.tu-berlin.de

Abstract

The new concept of *universal parameterized net classes* is introduced in order to allow a uniform approach to different kinds of Petri net classes. By different actualizations of the *net structure parameter* and the *data type parameter* we obtain several well-known net classes, like elementary nets, place-transition nets, coloured nets, predicate transition nets, and algebraic high-level nets, as well as several interesting new classes of low- and high-level nets. While the concept of *parameterized net classes* is defined on a purely set theoretical level, we introduce the extended concept of *universal parameterized net classes* taking into account also morphisms and universal properties in the sense of category theory. The extended concept leads to a uniform theory of constructions and compatibility results concerning union and fusion of nets for different types of net classes.

1 Introduction

Petri nets have been used sucessfully for more than three decades to model concurrent processes and distributed systems. Various kinds of Petri net classes with numerous features and analysis methods have been proposed in literature (among many others [Bra80, BRR87a, BRR87b]) for different purposes and application areas. The fact that Petri nets are widely used and are still considered to be an important topic in research, shows the usefulness and the power of this formalism. Nevertheless, the situation in the field of Petri nets is far from satisfactory, partly due to the enormous interest in Petri nets, that has been leading to a vast accumulation of dissimilar approaches. The different notions, definitions and techniques, both in literature and practice, make it hard to find a common understanding and to provide good reasons for the practical use of the nets. Moreover, the unstructured variety of Petri net approaches causes the new formulation and examination of similar concepts. The relation of these concepts requires complicated, but boring conversions. Most of the different concepts for Petri nets are defined explicitly for a single net class, although many of these notions are essentially the same for different kinds of net classes. Unfortunately, however, there is no abstract notion of Petri nets up to now, that permits the abstract formulation of such notions for a large variety of different net classes. The uniform approach to Petri nets, based on universal parametrized net classes, captures the common components of different kinds of Petri nets like places, transitions, net structure, and, in case of high-level nets, a data type part. Moreover, this approach treats low- and high-level nets in the same way, considering a trivial data type for low-level nets.

[*]A short version of this paper can be found in [EP97]

General notions, like firing behaviour, that are essential for all kinds of Petri nets, low- as well as high-level, are formulated within the frame of abstract Petri nets independently of their specific definition within a fixed net class. We do not consider this uniform approach as a net formalism for application purposes. Nevertheless, such an approach allows the easy transfer of results between different Petri net formalisms and thus has an impact on Petri nets used in practice. Hence, this concept comprises many known and several new net classes as special cases and allows their mutual comparison. Results achieved within this frame can be generalized to several different net classes. This means notions and results are achieved without further effort in each of these net classes, provided the general assumptions have been verified for the specific instance.

This paper is organized as follows. First, we sketch a purely set theoretic description of this uniform approach, called parametrized net classes. This includes the definition of the net structure parameter and the data type parameter. Then we extend these parameters into a categorical frame that allows the construction of the marking graph, some horizontal and vertical structuring techniques and compatibility results. The conclusion summarizes the achieved results.

2 Parametrized Net Classes

The basic idea of this uniform approach to Petri nets is to identify two parameters, that describe each of the net classes entirely. In case of the usual Petri nets this is the net structure and in case of high-level nets it is the net structure and the data type formalism. We call these parameters net structure parameter and data type parameter. The instanciation of these parameters leads to different types of Petri nets, or more precisely Petri net classes. In this section we introduce parameters for net classes, parametrized net classes and their instances leading to several well-known and some interesting new net classes.

2.1 Relevance of Parameters for Net Classes

The net structure parameter describes different low-level net classes. Several different net classes have been proposed over the last 30 years, see for example [Bra84, Rei85, RT86]. Moreover, the developments in software industry have yielded quite a large amount of variants that are equipped with additional features and/or restrictions. We propose an abstraction of net structure that can be instanciated to several net classes, including place/transition nets, elementary nets, variants of these and S-graphs. We have shown in [Pad96] that the underlying construction is general enough to comprise several different kinds of low-level nets and their net structure. The data type parameter is necessary, because several data type formalisms have been integrated with Petri nets leading to different notions of high-level nets. Typical examples are: indexed sets with place/transition nets leading to coloured nets [Jen81], predicate logic with elementary nets leading to predicate/transition nets [GL81], algebraic specifications with place/transition nets leading to different versions of algebraic high-level nets [Vau86, Rei91], ML with place/transition nets leading to coloured nets [Jen92], OBJ2 with superposed automata nets leading to OBJSA-nets [BCM88], and algebraic specifications with the Petri Box Calculus [BDH92] leading to A-nets [KP95]. In practice, there are also other data type formalisms like entity/relationship diagrams, SADT and many other semi-formal techniques that are combined with Petri nets in an informal way.

2.2 Algebraic Presentation of Place/Transition Nets

We use the algebraic presentation of Petri nets, that uses functions to relate a transition with its pre- and postdomain. This approach relates a transition on the one hand with all those places in its predomain using the function *pre* and on the other hand with its postdomain using the function *post*. In this algebraic presentation a place/transition net N is simply given by a 4-tuple $N = (P, T, pre, post)$ where P and T are the sets for places and transitions respectively and *pre* and *post* are functions $pre, post : T \to P^*$ from T to the free commutative monoid P^* over P. This construction is similar to the construction of words over some given alphabet. Due to the axiom of commutativity the elements of the free commutative monoid are considered to be linear sums over P, that is for each $t \in T$ we have $pre(t) = \sum_{i=1}^{k} n_i p_i$ for $p_i \in P$ and $n_i \in \mathbb{N}$ for some $k \in \mathbb{N}$. Note, that this is just the same as multisets. The marking is given by some element $m \in P^*$ and the operations for the computation of the firing behaviour are comparison, subtraction and addition based on linear sums, defined over the monoid operation.

This algebraic presentation [MM90] is equivalent to the classical presentation (see e.g. [Rei85, ER96]), but has the advantage of a clear and axiomatic presentation, thus it is much simpler to generalize.

2.3 Algebraic Presentation of Elementary Nets

In the case of elementary nets[1] the algebraic presentation is given by the power set construction $\mathcal{P}(P)$, that is $pre, post : T \to \mathcal{P}(P)$, because ${}^\bullet t = pre(t)$ and $t^\bullet = post(t)$ are given by subsets of P. Moreover, each element $m \in \mathcal{P}(P)$ can be considered as a marking of the elementary net. The firing behaviour makes use of the order on sets and the operations union and complement of sets.

2.4 Variants of Net Classes

Note, that there are several variants of place/transition nets, and similar for other types of nets, where nets are considered with initial marking or with labels on places, transitions, and/or arcs. However, in this paper we only consider a basic variant without initial markings and without labels. The neglect of the initial marking is due to our focus on structural composition techniques. The composition of nets without initial marking yields techniques as union and fusion. These techniques are independent from the behaviour of the net. The composition of nets with initial marking yields techniques as substitution and invocation, where the composition is dependent from the behaviour of the net. In the case of high-level nets our basic variant means in addition that a net includes one explicit model with total operations (resp. predicates) and that there are no firing conditions for the transitions (unguarded case). In our paper [ER96] we discuss several kinds of variants of algebraic high-level nets, which can also be extended to variants of other types of high-level nets considered in this paper.

2.5 Mathematical Notation of Parameters for Net Classes

In the following we distinguish between formal and actual parameters for the net structure and data type formalisms. The actual net structure parameter for a net class is based on the algebraic presentation of nets, more precisely the codomain of the pre- and postdomain functions. The algebraic presentation of different kinds of Petri nets as the actual parameter

[1]We talk about elementary nets as elementary (net) systems without an initial marking.

allows the generalisation in an axiomatic way, that is the formal parameter. Hence, it is the basic construction in order to express the difference of an actual and a formal parameter in an uniform approach to Petri nets.

For place/transition nets (in subsection 2.2) the codomain uses the construction of the free commutative monoid P^* over the set of places P. For elementary nets (in subsection 2.3) the power set construction is used. The calculation with markings is based on the operations union of $\mathcal{P}(P)$ and addition of P^* respectively. In order to generalize this computation a semigroup structure is employed in both classes. Hence, the constructions $\mathcal{P}(P)$ and P^* for each set P can be considered as functions from the class **Sets** of all sets via some class **Struct** of semigroups to the class **Sets**. These constructions are used as the actual parameter Net for the parameterized net classes. We consider $\mathcal{P}(P)$ and P^* as sets. The use of sets instead of semigroups allows the mapping from the transitions to these sets. Moreover, this has the advantage to consider nets, where the structure of the marking is different from the structure of the pre- and postdomain of the transitions, as for example in S-graphs, where markings contain multiple tokens, but the arc weight always equals one (see example 2.5.2.3). This motivates that in general an actual net structure parameter for a net class can be considered as the composition of two functions: $Net : \mathbf{Sets} \xrightarrow{F} \mathbf{Struct} \xrightarrow{G} \mathbf{Sets}$.

Based on the function Net we can describe Petri nets uniformly by $pre, post : T \to Net(P)$, where the specific net class depends on the choice of the function Net. Then $F(P)$ denotes the markings and the pre- and postdomain of the transitions, $F(T)$ yields transition vectors, and G relates the used construction (i.e. free monoids, power sets) with sets.

For high-level net classes we use the notion of institutions (see [GB84, ST84]), which is well-established in the area of abstract data types. Institutions are an abstract description of data type formalisms and generalize different formalisms, as algebraic specifications, predicate logic, functional programming languages, and so on. The basic idea is to assume axiomaticly some specification $SPEC$ and a class of models $\mathbf{Mod(SPEC)}$. Based on this theory an actual data type parameter for a net class consists of a class $\mathbf{SPEC}$ of data type specifications and for each $SPEC \in \mathbf{SPEC}$ a class $\mathbf{Mod(SPEC)}$ of models satisfying the specification $SPEC$. Hence, it can be represented by a function $Mod : \mathbf{SPEC} \to \mathbf{ModelClasses}$ where **ModelClasses** is the (super)class of all model classes.

Definition 2.5.1 (Formal Parameters of Net Classes)
The formal net structure parameter, respectively formal data type parameter, of a parametrized net class is given a pair of functions $Net : \mathbf{Sets} \xrightarrow{F} \mathbf{Struct} \xrightarrow{G} \mathbf{Sets}$ and a function $Mod : \mathbf{SPEC} \to \mathbf{ModelClasses}$ respectively, as motivated above.

Example 2.5.2 (Actual Parameters of Net Classes)

1. The free commutative monoid P^* over a set P defines the two functions $(\_)^* : \mathbf{Sets} \to \mathbf{Struct} \to \mathbf{Sets}$, where P^* togehter with the addition operation on linear sums is a semigroup (without this operation P^* is a set, thus the funcion G simply "forgets" the addition), which is the actual net structure parameter for the class of place/transition nets. The free commutative monoid P^* can be represented by formal sums $P^* = \{\sum_{i=1}^{k} n_i * p_i \mid p_i \in P,\ n_i \in \mathbb{N}\}$ with componentwise addition.

2. The powerset construction $\mathcal{P}(P)$ over a set P defines two functions $\mathcal{P} : \mathbf{Sets} \to \mathbf{Struct} \to \mathbf{Sets}$, where $\mathcal{P}(P)$ with union operation is a semi-group (without the union operation $\mathcal{P}(P)$ is a set), which is the actual net structure parameter for the class of elementary nets.

3. The actual net structure parameter for the subclass of place/transition nets, called S-Graphs (see [RT86]), where each transition has exactly one place in its pre- and post-domain respectively makes use of the compositionallity of the function Net. The corresponding net can be considered as graph, where the nodes are the places and the edges are the transitions, that is transitions are mapped to places by $pre, post : T \to P$. Nevertheless, markings are elements $m \in P^*$ (as usual in place/transition nets) rather than $m \in P$, which would allow only one token at all in the net, thus the intermediate construction has to be the free commutative monoid P^*. This is expressed by the pair of functions $SG : \mathbf{Sets} \to \mathbf{Struct} \to \mathbf{Sets}$, defined by $SG : X \mapsto X^* \mapsto X$ for each set X.

4. Let **SPEC** be the class of all algebraic specifications $SPEC = (S, OP, E)$ with signature (S, OP) and equations E and $\mathbf{Alg}(\mathbf{SPEC})$ the class of all $SPEC$-algebras A (see [EM85]). Then we obtain a function $Alg : \mathbf{SPEC} \to \mathbf{ModelClasses}$ which is the actual data type parameter for the class of algebraic high-level nets ([PER95]).

5. Let **FOSPEC** be the class of all first order predicate logic specifications $FOSPEC = (\Omega, \Pi, AXIOMS)$ with the signature (Ω, Π) and $AXIOMS$ being a set of closed formulas, and $\mathbf{FOMod}(\mathbf{FOSPEC})$ the class of all non-empty models satisfying the formulas in $AXIOMS$. Then we obtain a function $FOMod : \mathbf{FOSPEC} \to \mathbf{ModelClasses}$, which is the actual data type parameter for the class of predicate/transition nets ([GL81]).

6. As a uniform approach to Petri nets should comprise low- as well as high-level nets, we consider low-level nets as a special case of high-level nets with a data type that yields only one data element, the usual black token. This is merely a technical extension, because in [Pad96] it has been shown, that these high-level nets with a trivial data type correspond one-to-one to the usual low-level nets. The great advantage is that this conception allows to consider low- and high-level nets within one uniform approach. In order to be able to define low-level nets as special case of high-level nets we define the following trivial actual data type parameter $Triv : \mathbf{TRIV} \to \mathbf{ModelClasses}$, where the class **TRIV** consists only of one element called trivial specification $TRIV$, and $\mathbf{Triv}(\mathbf{TRIV})$ is the model class, consisting only of one model, the one-element set $\{\bullet\}$, representing a single token.

7. There are also actual data type parameters for the class of coloured nets in the sense of [Jen81] and [Jen92], which are based on indexed sets and the functional language ML, respectively (see [Pad96]).

Note, that the theory of institutions allows to treat different data type descriptions in the same way, but does not neglect the differences. This is due to the abstract formulation of this theory. Our uniform approach to Petri nets is motivated by this abstraction.

Now we are able to define parameterized net classes and their instantiations mentioned above.

Though parametrized net classes cannot yield concrete nets, as the formal parameters are not yet actualized, they give rise to net patterns. Net patterns constitute a pattern for Petri nets consisting of places, transitions, pre- and postdomain, specification and a data type model. Nevertheless, neither the structure of pre- and postdomain is fixed – due to the net parameter – nor the kind of the specification and its model – due to the data type parameter.

Definition 2.5.3 (Parameterized Net Classes)

A parameterized net class is defined by a formal net structure parameter (Net, Mod) with $Net = G \circ F$ and $Mod : \mathbf{SPEC} \to \mathbf{ModelClasses}$ (see 2.5.1) consists of all net patterns $N = (P, T, SPEC, A, pre, post)$ satisfying the following conditions:

- P and T are sets of places and transitions respectively,
- $SPEC \in \mathbf{SPEC}$ is the data type specification
- $A \in \mathbf{Mod}(\mathbf{SPEC})$, called data type model
- $pre, post : T \to Net(T_{SPEC} \times P)$ are the pre- and postdomain functions, where T_{SPEC} is a distinguished model with respect to the specification $SPEC$ (e.g. where the elements are congruence classes of terms over the specification $SPEC$).

In the case of low-level nets we have $Mod = Triv$ (see 2.5.2.6) and hence $\mathbf{SPEC} = \mathbf{TRIV}$ and $A = \{\bullet\}$ which are omitted as components of a net. Since T_{SPEC} consists of a single element, we obtain $N = (P, T, pre, post)$ with $pre, post : T \to Net(P)$

Remark 2.5.4
Other parameterized and actual net classes can be defined by other variants of net classes discussed in example 2.5.2. In our notion above we only consider the basic variant without initial markings, without labels, with only one explicit (total) model and without firing conditions for transitions.

The behaviour can be given already for the net patterns, although the net pattern is no "real net" unless the formal parameters are actualized. The behaviour of net patterns yields a uniform description of the behaviour in different Petri net classes:

The firing of a transition vector can be defined axiomaticly using the pair of functions $Net : \mathbf{Sets} \xrightarrow{F} \mathbf{Struct} \xrightarrow{G} \mathbf{Sets}$, the operation $+$, given by the semigroup structure, and the extensions $\overline{pre}$ (resp. $\overline{post}$) of pre (resp. $post$) to the semigroup structure. The marking is given by $m \in F(P)$. A transition vector $v \in F(T)$ is enabled under $m \in F(P)$ if there exists $\widehat{m} \in F(P)$ so that $m = \widehat{m} + \overline{pre}(v)$. The follower marking $m' \in F(P)$ obtained by firing v under m is given by $m' = \widehat{m} + \overline{post}(v)$

The firing of high-level nets involves additionally the assignment of values to variables. This can be done nicely using notions from the theory of institution, but exceeds the frame of this paper (see [Pad96]).

Example 2.5.5 (Actual Net Classes)
A survey of actual net classes defined by the actual parameter given in example 2.5.2 is given in the table below, where well-known net classes are shown as explicit entries in the table while each blank entry corresponds to a new or less well-known net class.

Mod / *Net*	Triv	Alg. Spec.	Pred. Logic	ML	Indexed Sets
powerset	Elem. Nets [RT86]		PrT-Nets [GL81]		
monoid	P/T-Nets [Rei85]	AHL-Nets [PER95]		CPN '91 [Jen92]	CPN '82 [Jen81]
monoid identity	S-Graph [RT86]				
...	...				

In more detail we have:

- Place/Transition nets defined by $Net = (\_)^*$ (see 2.5.2.1) are given by $N = (P, T, pre, post)$ with $pre, post : T \to P^*$ (see 2.2, 2.5).
- Elementary nets defined by $Net = \mathcal{P}$ (see 2.5.2.2) are given by $N = (P, T, pre, post)$ with $pre, post : T \to \mathcal{P}(P)$ (see 2.5).
- S-graphs defined by $Net = SG$ (see 2.5.2.3) are given by $N = (P, T, pre, post)$ with $pre, post : T \to P$.
- AHL-nets defined by $Net = (\_)^*$ and $Mod = Alg$ (see 2.5.2.4) are given by $N = (P, T, SPEC, A, pre, post)$ with $pre, post : T \to (T_{OP}(X) \times P)^*$. Here the distinguished model is the quotient term algebra $T_{OP}(X)$, that is $T_{OP}(X)$ are the terms with variables X over $SPEC$.
- For a presentation of predicate/transition nets and different versions of coloured nets in our framework see [Pad96].

3 Universal Parametrized Net Classes

The main idea of universal parameterized net classes is to extend the notion of parameterized net classes studied in the previous section by morphisms on different levels. Hence, the classes become categories and the corresponding functions become functors in the sense of category theory [AHS90]. In other words the set theoretical approach of parameterized net classes becomes a categorical approach of universal parameterized net classes. Especially, we obtain also morphisms of nets for all types of net classes. Since morphisms of nets are rarely used for Petri nets up to now we motivate the benefits of morphisms for Petri nets. For this purpose we first review net morphisms for place/transition nets in algebraic presentation (see subsection 2.2). Then we discuss the benefit of morphisms in this case which is also valid for other types of net classes, and we give a short outline how to extend section 2 to universal parameterized net classes. Finally, we discuss general constructions and results which have been obtained in this framework.

3.1 Categorical Concepts

Category theory is a universal formalism which is successfully used in several fields of mathematics and theoretical computer science. It has been developed for about 50 years and its influence can be found in most branches of structural mathematics and, for about 25 years in several areas of theoretical computer science. In the survey [EGW96] it has been shown that the following areas in computer science have been influenced by category theory: Automata and system theory, flow charts and recursion, λ-calculus and functional languages, algebraic specifications, logical systems and type theory, graph transformation, Petri nets and replacement systems. The aim of category theory is to present structural dependencies and universal notions that can be found in many (mathematical) areas and to give a uniform frame independently of internal structures. This uniform frame and the universality of the concepts distinguish category theory as a common language for the modeling and analysis of complex structures, as well as for a unified view of the development of theories, and for the integration of different theories within computer science. The main purpose of category theory is to have a uniform frame for different kinds of mathematical structures, mappings between structures, and constructions of structures. The most fundamental notions in category theory are on the one hand *categories* consisting of *objects* and *morphisms* and on the other hand *functors* defining structure compatible mappings between categories. Another important concept of category theory is that of limits and colimits, especially product, equalizers, and pullbacks and the dual concepts of coproducts, coequalizers and pushouts. In [Pad96] we especially need pushouts and coequalizers corresponding to a union of objects with shared subobjects and the fusion of subobjects, respectively. Universal properties express that these constructs are generated. These universal properties imply that the corresponding construction is essentially unique. They are strongly exploited for the results we obtain in [Pad96]. Special colimits as pushouts and coequalizers are the basis for the just mentioned structuring techniques. Injective pullbacks correspond to the intersection of nets. Furthermore, the notions and results for rule-based modification depend on these constructions. The marking graph construction and the realization construction are derived from the net structure, and the preservation of colimits by free constructions yields the compatibitlity of the marking graph construction with the structuring techniques (see [Pad96]).

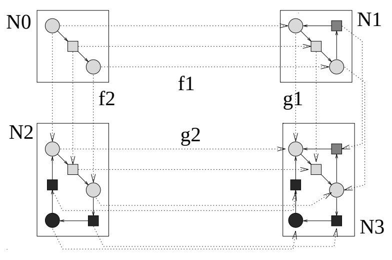
3.2 Morphisms of Place/Transition Nets

Given two place/transition nets $N1$ and $N2$ with $N_i = (P_i, T_i, pre_i, post_i)$ for $(i = 1,2)$ (see algebraic presentation in 2.2) a morphism $f : N_1 \to N_2$ of place/transition nets is a pair $f = (f_P : P_1 \to P_2, f_T : T_1 \to T_2)$ of functions, such that we have $f_P^* \circ pre_1 = pre_2 \circ f_T$ and $f_P^* \circ post_1 = post_2 \circ f_T$ where $f_P^* : P_1^* \to P_2^*$ is defined by $f_P^*(\Sigma_{i=1}^k n_i * p_i) = \Sigma_{i=1}^k n_i f_P(p_i)$. This means that the following diagram commutes for pre- and postdomain functions respectively.

$$\begin{tikzcd}
T_1 \arrow[r, shift left, "pre_1"] \arrow[r, shift right, "post_1"'] \arrow[d, "f_T"'] & P_1^* \arrow[d, "f_P^*"] \\
T_2 \arrow[r, shift left, "pre_2"] \arrow[r, shift right, "post_2"'] & P_2^*
\end{tikzcd}$$

Example 3.2.1 (Example of union in low-level nets)

This example illustrates the union of place/transition nets of N_1, and N_2 with the interface N_0 resulting in N_3 (that is $N_3 = N_1 +_{N_0} N_2$). The morphisms $f_1 : N_0 \to N_1$, $f_2 : N_0 \to N_1$, $g_1 : N_1 \to N_3$, and $g_2 : N_2 \to N_1$ are denoted explicitly by the dotted lines.



3.3 Benefits of Morphisms for Petri Nets

Similar to subsection 3.2 for place/transition nets morphisms can also be defined for all other kinds of Petri nets, including low-level and high-level nets. The main benefits – illustrated using place/transition nets – are the following:

1. A morphism $f : N_1 \to N_2$ of nets allows to express the structural relationship between nets N_1 and N_2. If f is injective (in all components) then N_1 can be considered as a subset of N_2.

   In general f may map different places p_1 and p_1' or transitions t_1 and t_1' of N_1 to only one place p_2 or one transition t_2 of N_2. Then only a homomorphism image of N_1 is a subnet of N_2. In fact, there may be different morphisms $f, g : N_1 \to N_2$ which corresponds to different occurrences of the net N_1 in the net N_2. In example 3.2.1 all morphisms are injective such that N_0 can be considered as subnet of N_1 via f_1 and of N_2 via f_2. Moreover, N_1 and N_2 can be considered as subnet of N_3 via g_1 and g_2 respectively.

2. A bijective morphism $f : N_1 \xrightarrow{\sim} N_2$ is called isomorphism. In this case the nets N_1 and N_2 are called isomorphic, written $N_1 \cong N_2$, which means that they are equal up to renaming of places and transitions.

3. The composition of net morphisms $f_1 : N_1 \rightarrow N_2$ and $f_2 : N_2 \rightarrow N_3$ is again a net morphism $f_2 \circ f_1 : N_1 \rightarrow N_3$ (see also $g_1 \circ f_1 = g_2 \circ f_2 : N_0 \rightarrow N_3$ in 3.2.1). Moreover, this composition is associative and for each net N there is an identity morphism $id_N : N \rightarrow N$ such that we have $f_1 \circ id_{N_1} = f_1$ and $id_{N_2} \circ f_1 = f_1$ for all $f_1 : N_1 \rightarrow N_2$. This means that the class of all nets (of a given type TYP) together with all net morphisms constitutes a category **TypNet**.

   This allows to apply constructions and results from category theory to different types of nets and net classes. Note, that each pair (Net, Mod) of actual parameters defines a type $TYP = (Net, Mod)$ and hence an actual net class (see 2.5.5) which is the object class of the category **TypNet**.

4. Morphisms can be used to define the horizontal structuring of nets, for example the net N_3 in 3.2.1 as union of N_1 and N_2 via the common subnet N_0. Vice versa, the nets N_1 and N_2 with subnet N_0 (distinguished by morphisms $f_1 : N_0 \rightarrow N_1$ and $f_2 : N_0 \rightarrow N_2$) can be composed leading to net $N_3 = N_1 +_{N_0} N_2$. In fact, this union of nets is also a pushout construction in the category **TypNet**. This allows to apply general results of category theory like composition and decomposition properties of pushouts to the union construction of nets, for example associativity and commutativity of union up to isomorphism.

5. Morphisms can also be used to define refinement of nets. In several cases more general morphisms than those in 3.2 should be considered for this purpose. One simple generalization of 3.2 is to replace $f_P^* : P_1^* \rightarrow P_2^*$ generated by $f_P : P_1 \rightarrow P_2$ by an arbitrary monoid homomorphism $\hat{f}_P : P_1^* \rightarrow P_2^*$. This allows to map one place p_1 in P_1 to a sum of places, that is $p_1 \mapsto \Sigma_{i=1}^k n_i * p_{2_i}$ for $p_{2_i} \in P_2$, which is important for refinement.

6. A morphism $f : N_1 \rightarrow N_2$ of place/transition nets preserves the firing behaviour: If transition vector $v \in F(T)$ is enabled under marking m in net N_1 leading to marking m', that is $m[v > m'$ then also the transition $f_T(v)$ is enabled under marking $f_P^*(m)$ in net N_2 leading to marking $f_P^*(m')$, that is $f_P^*(m)[f_T(v) > f_P^*(m')$. In a similar way morphisms preserve the firing behaviour also for other types of nets. Specific kinds of net morphisms can be considered to preserve other kinds of Petri net properties, for example deadlock-freeness. Especially, isomorphisms preserve all kinds of net properties which do not depend on a specific notation.

3.4 Universal Parameters of Net Classes

The parameters of net classes considered in 2.5.1 become universal parameters of net classes, if all classes and functions are replaced by categories and functors, respectively. In more detail the classes of sets, structures, specifications, and model classes are extended by suitable morphisms leading to categories **Sets** of sets, **Struct** of structures, **SPEC** of specifications, and **ModelClasses** of model-classes, and for each $SPEC \in$ **SPEC** a category **Mod**(**SPEC**) of $SPEC$-models. Moreover, the functions are extended to become functors: $Net : Sets \xrightarrow{F} Struct \xrightarrow{G} Sets$, the universal net structure parameter and $Mod : \mathbf{SPEC}^{op} \rightarrow$ **ModelClasses**, the universal data type parameter, where $Net = G \circ F$ and the functor F is a *free functor* with respect to the *forgetful functor* G, and Mod is a contravariant functor from **SPEC** to **ModelClasses** in the sense of category theory ([AHS90]). Note, that we use an overloaded notation, where **Sets**, **Struct**, **SPEC**, **ModelClasses**, and **Mod(SPEC)** denote classes and Net, F, G, and Mod denote functions in section 2, while they denote categories and functors respectively in this section. In fact, all the examples of actual parameters of net classes given in 2.5.2 can be extended to universal parameters of net classes with well-known categories and functors (see [Pad96]). We only consider the universal net structure parameter of place/transition nets in more detail (see 2.5.2.1):

Let $\mathbf{Struct} = \mathbf{CMon}$ be the category of commutative monoids, $F : \mathbf{Sets} \to \mathbf{CMon}$ the free commutative monoid construction, that is $F(P) = (P^*, 0, +)$, $G : \mathbf{CMon} \to \mathbf{Sets}$ the forgetful functor, defined by $G(M, \epsilon, \circ) = M$, forgetting only about the neutral element ϵ and the monoid operator $\circ$. Then $Net : \mathbf{Sets} \xrightarrow{F} \mathbf{CMon} \xrightarrow{G} \mathbf{Sets}$ with $Net(P) = P^*$ is the universal net structure functor for the class of place/transition nets.

In fact, F is a free functor with respect to G, because for each set P the free construction $F(P) = (P^*, 0, +)$ together with the inclusion $u_P : P \to G \circ F(P) = P^*$ satisfies the following universal property: For each commutative monoid $(M, \epsilon, \circ)$ and each function $f : P \to G(M, \epsilon, \circ) = M$ there is a unique monoid homomorphism $\overline{f} : F(P) = (P^*, 0, +) \to (M, \epsilon, \circ)$ such that $G(\overline{f}) \circ u_P = f$:

$$\begin{tikzcd}
P \arrow[r, "f"] \arrow[d, "u_P"'] & G((M,\epsilon,\circ) = M \\
G(F(P)) \arrow[ur, "G(\ \overline{f} : F(P) \to (M,\epsilon,\circ)\ )"'] &
\end{tikzcd}$$

In fact, $\overline{f} : (P^*, 0, +) \to (M, \epsilon, \circ)$ is uniquely defined by $\overline{f}(0) = \epsilon$ and $\overline{f}(\ \sum_{i=1}^{k} n_i * p_i\ ) = \Pi_{i=1}^{k} n_i * f(p_i)$. This universal property allows to extend the pre- and postdomain functions of place/transition nets $pre, post : T \to P^*$ – and similar for other types of nets – to monoid homomorphisms $\overline{pre}, \overline{post} : T^* \to P^*$ and hence to parallel firing of transitions.

3.5 Universal Parameterized Net Classes

Parameterized classes as considered in 2.5.1 become universal parameterized classes, if we replace the actual parameters (Net, Mod), which are functions, by universal parameters (Net, Mod), which are functors (see subsection 3.4). In this case we have in addition to the net patterns $N = (P, T, SPEC, A, pre, post)$ of the corresponding parametrized net class also net pattern morphisms $f : N1 \to N2$ leading to a category **TypNet** for the type $TYP = (Net, Mod)$. In the case of low-level net patterns $N_i = (P_i, T_i, pre_i, post_i)$ for $(i = 1, 2)$ of type (Net, Mod) a net pattern morphism $f : N_1 \to N_2$ is a pair of functions $f = (f_P : P_1 \to P_2, f_T : T_1 \to T_2)$ such that $pre_2 \circ f_T = Net(f_P) \circ pre_1$ and $post_2 \circ f_T = Net(f_P) \circ post_1$. In the special case $Net = (\_)^*$ we obtain the notion of morphisms for place/transition nets (see subsection 3.2). All the examples of actual net classes given in 2.8 can be extended to examples of universal actual net classes defined by the corresponding universal parameters (see subsection 3.4). Moreover, these morphisms preserve the firing behaviour of Petri nets.

3.6 Uniform Constructions and Results

In section 2 we have shown how to obtain several well-known and new net classes in a uniform way by the notion of parameterized and actual net classes. Now we raise the question, how far it is possible to obtain well-known results for each of these net classes in a uniform way. At first sight this seems hopeless, because each type of Petri net has its own notation and own kind of problems, although the general idea of most constructions and results is quite similar. However, the presentation of net classes as instances of parameterized net classes opens the way to study the theory on the level of parameterized net classes rather than for specific actual ones. In fact, this has been done to a certain extend in the Ph.D. thesis [Pad96] using the notion of *Abstract Petri Nets*, which corresponds to the notion of *Universal Parameterized Net Classes* in this paper. In the following we summarize some main constructions and results for abstract Petri nets in the terminology of this paper. In this way we obtain uniform constructions and results for all the actual net classes (see 2.5.5) which are instanciations of universal parameterized net classes:

- There is a uniform notion of net patterns, marking of net patterns, enabling and firing of transitions.
- Morphisms preserve the firing of transitions.
- Firing of transitions can be extended to parallel and concurrent firing in a uniform way (see subsection 3.3).
- In the case of low level nets there is a uniform construction of the marking graph of a net in terms of F-graphs and a characterization of all those F-graphs, which are realizable by nets in the net class defined by $Net = G \circ F$.
- There is a uniform construction of the operations *union* and *fusion* for nets in the sense of [Jen92], which are most important for horizontal structuring of nets.
- Refinement is an essential technique for vertical structuring of the software development process. Several refinement notions are known in the Petri net literature (see for example [BGV90]). Rule-based modification comprises these in a uniform way using ideas from the theory of graph grammars and high-level replacement systems [Pad96]. Moreover, important results concerning independence, parallelism and concurrency of rule-based modification – developed first in the theory of graph grammars – have been extended to universal parameterized net classes.
- Horizontal structuring of nets based on union and fusion is compatible with rule-based modification of nets, provided that certain independence conditions are satisfied.
- There is a uniform construction of flattening from high-level nets of type $TYP = (Net, Mod)$ to low-level nets of type $TYP = (Net, Triv)$.
- There is a uniform notion of morphisms for nets of type $TYP = (Net, Mod)$ leading to a category **TypNet** of nets of type TYP. The category **TypNet** is cocomplete, which includes as special cases existence and construction of pushouts and coequalizers corresponding to union and fusion of nets.

Finally, let us point out that for most of the uniform constructions and results above it seems to be necessary to consider morphisms of nets (see 3.3) This means that it is worthwhile to consider universal parameterized net classes including a few notions of category theory and not only parameterized net classes based on set theoretical notions.

4 Conclusion

In this paper we have given a uniform approach to different types of Petri nets, including low-level nets, like elementary and place/transition nets, as well as high-level nets, like algebraic high-level or predicate/transition nets. The main idea of this approach is a first step to introduce parameterized net classes, which are based on a net structure parameter for low-level nets and in addition a data type parameter for high-level nets. By instanciation of these parameters we obtain many well-known net classes studied in the literature but also several new interesting net classes in a uniform way. In a second step we extend the parameters of net classes to universal parameters leading to the notion of universal parameterized and actual net classes. The main idea of this extension is to study not only classes of specifications and nets, but to consider also suitable morphisms leading to different kinds of categories. The concept of universal parameterized net classes allows to obtain several constructions and results known for specific low-level or high-level nets in a uniform way for all net classes which can be obtained by suitable instantiations of the parameters. A formal version of these constructions and results is presented in the theory of abstract Petri nets studied in [Pad96]. This, however, is only a first step towards a general uniform theory for different types of Petri nets.

References

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

[BCM88] E. Battiston, F. De Cindio, and G. Mauri, *OBJSA nets: a class of high-level nets having objects as domains*, Advances in Petri nets (G. Rozenberg, ed.), vol. 340, Springer Verlag Berlin, 1988, pp. 20–43.

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

[BGV90] W. Brauer, R. Gold, and W. Vogler, *A Survey of Behaviour and Equivalence Preserving Refinements of Petri Nets*, Advances in Petri Nets, LNCS 483 (1990).

[Bra80] W. Brauer (ed.), *Net Theory and Aplications*, Springer, LNCS 84, 1980.

[Bra84] W. Brauer, *How to play the token game*, Petri Net Newsletter **16** (1984), 3–13.

[BRR87a] W. Brauer, W. Reisig, and G. Rozenberg (eds.), *Petri Nets: Applications and Relations to Other Models of Concurrency*, Springer, LNCS 255, 1987.

[BRR87b] W. Brauer, W. Reisig, and G. Rozenberg (eds.), *Petri Nets: Central Models and Their Properties*, Springer, LNCS 254, 1987.

[EGW96] H. Ehrig, M. Große-Rhode, and Uwe Wolter, *On the role of category theory in the area of algebraic specifications*, LNCS , Proc. WADT11, Oslo, 1996.

[EM85] H. Ehrig and B. Mahr, *Fundamentals of algebraic specifications 1: Equations and initial semantics*, EACTS Monographs on Theoretical Computer Science, vol. 6, Berlin, 1985.

[EP97] H. Ehrig and J. Padberg, *A Uniform Approach to Petri Nets*, Springer, 1997, To appear.

[ER96] H. Ehrig and W. Reisig, *Integration of Algebraic Specifications and Petri Nets*, Bulletin EATCS, Formal Specification Column (1996), no. 61, 52–58.

[GB84] J.A. Goguen and R.M. Burstall, *Introducing institutions*, Proc. Logics of Programming Workshop, Carnegie-Mellon Springer LNCS 164 (1984), 221 – 256.

[GL81] H.J. Genrich and K. Lautenbach, *System modelling with high-level Petri nets*, Theoretical Computer Science 13 (1981),pp. 109–136.

[Jen81] K. Jensen, *Coloured petri nets and the invariant method*, Theoretical Computer Science 14 (1981),pp. 317–336.

[Jen92] K. Jensen, *Coloured Petri nets. basic concepts, analysis methods and practical use*, vol. 1, Springer, 1992.

[KP95] H. Klaudel and E. Pelz, *Communication as unification in the Petri Box Calculus*, Tech. report, LRI, Universite de Paris Sud, 1995.

[MM90] J. Meseguer and U. Montanari, *Petri nets are monoids*, Information and Computation **88** (1990), no. 2, 105–155.

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

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro, *Algebraic high-level net transformation systems*, Mathematical Structures in Computer Science **5** (1995), 217–256.

[Rei85] W. Reisig, *Petri nets*, EATCS Monographs on Theoretical Computer Science, vol. 4, Springer-Verlag, 1985.

[Rei91] W. Reisig, *Petri nets and abstract data types*, Theoretical Computer Science **80** (1991), 1 – 34 (fundamental studies).

[RT86] G. Rozenberg and P.S. Thiagarajan, *Petri nets: Basic notions, structure, behaviour*, Current Trends in Concurrency, 1986, 224, pp. 585–668.

[ST84] D. T. Sannella and A. Tarlecki, *Building specifications in an arbitrary institution*, Proc. Int. Symposium on Semantics of Data Types, LNCS 173, Springer, 1984, pp. 337–356.

[Vau86] J. Vautherin, *Parallel specification with coloured Petri nets and algebraic data types*, Proc. of the 7th European Workshop on Application and Theory of Petri nets (Oxford, England), jul. 1986, pp. 5–23.