High-Level Net Processes

Hartmut Ehrig[1], Kathrin Hoffmann[1], Julia Padberg[1], Paolo Baldan[2], and Reiko Heckel[3]

[1] Technical University of Berlin, Germany
{ehrig,hoffmann,padberg}@cs.tu-berlin.de
[2] University of Pisa, Italy
baldan@di.unipi.it
[3] University of Paderborn, Germany
reiko@upb.de

Abstract. The notion of processes for low-level Petri nets based on occurrence nets is well known and it represents the basis for the study of the non-sequential behavior of Petri nets. Processes for high-level nets N are often defined as processes of the low level net $Flat(N)$ which is obtained from N via a construction called "flattening". In this paper we define high-level processes for high-level nets based on a suitable notion of high-level occurrence nets. The flattening of a high-level occurrence net is in general not a low-level occurrence net, due to so called "assignment conflicts" in the high-level net. The main technical result is a syntactical characterization of assignment conflicts. But the main focus of this paper is a conceptual discussion of future perspectives of high-level net processes including concurrency and data type aspects. Specifically, in the second part of the paper, we discuss possible extensions of high-level net processes, which are formally introduced for algebraic high-level nets in the first part of this paper. Of special interest are high-level processes with data type behavior, amalgamation, and other kinds of constructions, which are essential aspects for a proposed component concept for high-level nets.

1 Introduction

High-level nets are one of the most important examples of integrated data type and process modeling techniques for concurrent and distributed systems (see [EGH92, Jen92]). For low-level Petri nets the notion of nondeterministic and deterministic processes is an essential concept to capture their non-sequential truly concurrent behavior. Especially in the case of elementary net systems and safe place/transition nets this has been worked out in a fully satisfactory way by Rozenberg, Winskel, Nielsen, Goltz, Reisig, Degano, Meseguer, Montanari and other authors [NPW81, GR83, Roz87, Win87, Win88, DMM89, Eng91, MMS97] leading to different notions of deterministic and nondeterministic processes and to a truly concurrent semantics of Petri nets in terms of prime algebraic domains and event structures. The situation is already slightly more difficult in the case of non-safe place/transition nets. In order to capture a satisfactory notion of

W. Brauer et al. (Eds.): Formal and Natural Computing, LNCS 2300, pp. 191–219, 2002.

causality and conflict in the case of multiple tokens on one place, the dichotomy between the "individual token" and "collective token" views has been developed in the literature and coined as such by van Glabbeck and Plotkin in [vGP95]. On the basis of the individual token interpretation Meseguer, Montanari and Sassone [MMS97] have extended Winskel's adjunction between safe Petri nets and prime event structures to general place/transition nets. Since the flattening of high-level nets in general leads to non-safe place/transition nets, processes for general place/transition nets are a prerequisite to study the concurrent behavior of a high-level net N via the flattening $Flat(N)$ using the corresponding semantics and techniques of low-level nets. Especially, this allows to define processes of a high-level net N as low-level processes of the flattening $Flat(N)$ of N. This view of processes of high-level nets or an equivalent presentation has been mainly considered in the literature up to now.

In this paper we claim, however, that a more adequate view of processes for high-level nets should capture the distinction between the data processing and the concurrency aspect of processes in the sense of basic procedures for concurrent and distributed systems in software engineering and communication technology. For this purpose we propose a notion of high-level process for high-level nets which is defined independently of flattening.

The essential idea is to generalize the concept of occurrence net from the low-level to the high-level case. This means that the net structure of a high-level occurrence net has similar properties like a low-level occurrence net, like unitarity, conflict freeness, and acyclicity. But we drop the idea that an occurrence net captures essentially one concurrent computation. Instead, a high-level occurrence net and a high-level process are intended to capture a set of different concurrent computations corresponding to different input parameters of the process. This is in some sense analogous to procedures in programming languages, where a procedure works on a set of input parameters: Each instantiation or call of the procedure with suitable input parameters leads to a run of the procedure and hence to one specific computation. In fact, high-level processes can be considered to have a set of initial markings for the input places of the corresponding occurrence net, whereas there is only one implicit initial marking of the input places for low-level occurrence nets.

The paper is organized as follows. First in Section 2 we define high-level processes for algebraic high-level nets (AHL-nets) without initial marking. The case with initial markings is discussed in Section 4. In Section 3, we review the flattening of AHL-nets (see e.g. [EPR94]) and apply it to high-level processes. As expected, the flattening of a high-level process is in general not a low-level process, due to so called "assignment conflicts", which may occur in high-level occurrence nets. As main technical result we give a syntactical characterization of assignment conflicts. The main idea, however, is that a high-level process of a high-level net N corresponds to a set of low-level processes of the flattening $Flat(N)$ of N.

In Section 4, we discuss possible extensions of our basic notion of high-level processes. The first extension is that from algebraic-high level nets to that of

other types of high-level nets in the framework of parameterized net classes ([EP97a, EP97b, PE01]) and abstract Petri nets [Pad96]. Another extension is that from AHL-processes in Section 2 to open AHL-processes in the sense of open nets and open processes studied for Place/Transition nets most recently in the paper [BCEH01]. As mentioned already, another important extension is the case of AHL-processes with a set of initial markings, which also allows to define a data type behavior of high-level processes. This extension allows to consider AHL-nets and AHL-processes as an instantiation of the general concept of integrated data type and process modeling techniques in [EO01a] and leads to a component concept for AHL-nets with AHL-processes in the sense of [EO01b]. Last but not least we discuss in Section 4 how well-known constructions of low-level processes, like unfolding and concatenation can be extended to high-level processes. Moreover, we propose also new constructions for low-level and high-level processes via process terms based on constructions like sequential and parallel composition as well as amalgamation of processes in the sense of [BCEH01]. This should allow to define process types based on nets and net processes in analogy to data types [EM85] as advocated in [EMP97] already.

Acknowledgements

We are grateful to the referees of the paper for several useful comments and to Claudia Ermel and Maria Oswald for careful figure drawing and typing.

2 Algebraic High-Level Processes

In this section we review the concept of algebraic high-level net and we give a formal definition of algebraic high-level processes based on a suitable high-level notion of occurrence net. The well known example of dining philosophers in the high-level case (see [EPR94, PER95]) serves as running example.

The version of AHL-nets defined below is similar to [EPR94, PER95] but for the fact that places are typed, that is, the data elements on these places and the terms occurring in the inscriptions of the attached arcs are required to be of a specified sort. This typing reduces the set of possible markings and helps to keep the unfolding of a high-level net manageable.

Definition 2.1 (Algebraic High-Level Net).
An *algebraic high-level net* (*AHL-net*),

$$N = (SPEC, P, T, pre, post, cond, type, A)$$

consists of an algebraic specification $SPEC = (S, OP, E; X)$ with equations E and additional variables X over the signature (S, OP) (see [EM85]), sets P and T of places and transitions, respectively, pre- and post-domain functions

$$pre, post : T \to (T_{OP}(X) \otimes P)^{\oplus}$$

assigning to each transition $t \in T$ the pre- and post-domains $pre(t)$ and $post(t)$ (see Remark 1), respectively, a firing condition function

$$cond : T \to \mathcal{P}_{fin}(EQNS(S, OP, X))$$

assigning to each transition $t \in T$ a finite set $cond(t)$ of equations over the signature (S, OP) with variables X, a type function

$$type : P \to S$$

assigning to each place $p \in P$ a sort $type(p) \in S$, and an (S, OP, E)-algebra A (see [EM85]).

△

Remarks

1. Denoting by $T_{OP}(X)$ the set of terms with variables X over the signature (S, OP) (see [EM85]), and by $M^{\oplus}$ the free commutative monoid over a set M, the set of all type-consistent arc inscriptions $T_{OP}(X) \otimes P$ is defined by

   $$T_{OP}(X) \otimes P = \{(term, p) | term \in T_{OP}(X)_{type(p)}, p \in P\}.$$

   Thus, $pre(t)$ (and similar $post(t)$) is of the form $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ $(n \geq 0)$ with $p_i \in P, term_i \in T_{OP}(X)_{type(p_i)}$. This means, $\{p_1, \ldots, p_n\}$ is the pre-domain of t with arc-inscription $term_i$ for the arc from p_i to t if the $p_1, \ldots, p_n$ are pairwise distinct (unary case) and arc-inscription $term_{i1} \oplus \cdots \oplus term_{ik}$ for $p_{i1} = \cdots = p_{ik}$ (multi case).
2. The proposed version of AHL-nets is similar to [EPR94, PER95], but for the use of free commutative monoids $M^{\oplus}$ (cf. [MM90]) instead of free Abelian groups M^{ab}. In this paper the use of $M^{\oplus}$ is more suitable than M^{ab} in order to compare the high level with the classical low level case of place/transition nets. Moreover, as discussed above, places are typed. Therefore, the cartesian product $T_{OP}(X) \times P$ used in [PER95] has been replaced by its subset $T_{OP}(X) \otimes P$. Consequently a marking m is now an element $m \in (A \otimes P)^{\oplus}$ with

   $$A \otimes P = \{(a, p) | a \in A_{type(p)}, p \in P\}.$$

3. The case of AHL-nets and corresponding AHL-processes with initial markings $init \in (A \otimes P)^{\oplus}$ will be discussed in Section 4.

Enabling and firing of transitions are defined as follows.

Definition 2.2 (Algebraic High-Level Net).
Given an AHL-net as above and a transition $t \in T$, $Var(t)$ denotes the set of variables occurring in $pre(t), post(t)$, and $cond(t)$. An assignment $asg_A : Var(t) \to A$ is called consistent if the equations $cond(t)$ are satisfied in A under asg_A.

The marking $pre_A(t, asg_A)$ – and similarly $post_A(t, asg_A)$ – is defined for $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ by

$$pre_A(t, asg_A) = \sum_{i=1}^{n}(\overline{asg_A}(term_i), p_i),$$

where $\overline{asg_A} : T_{OP}(Var(t)) \to A$ is the extension of the assignment asg_A to an evaluation of terms (see [EM85]).

A transition $t \in T$ is enabled under a consistent assignment $asg_A : Var(t) \to A$ and marking $m \in (A \otimes P)^{\oplus}$, if

$$pre_A(t, asg_A) \leq m,$$

In this case, the successor marking m' is defined by

$$m' = m \ominus pre_A(t, asg_A) \oplus post_A(t, asg_A)$$

where $pre_A(t, asg_A)$ and similar $post_A(t, asg_A)$ is defined for $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ by

$$pre_A(t, asg_A) = \sum_{i=1}^{n}(\overline{asg_A}(term_i), p_i).$$

△

Example 2.3 (Dining Philosophers).
In analogy to [PER95] the AHL-net $DIPHI$ for dining philosophers (with n philosophers) is given in Figure 1 together with the algebraic specification

$\underline{\underline{diphi}} = \underline{sorts:}\ philo, fork$

$\underline{opns:}\ p_1, \ldots, p_n :\rightarrow philo$
$f_1, \ldots, f_n :\rightarrow fork$
$ls : philo \to fork$
$rs : philo \to fork$

$\underline{eqns:}\ ls(p_1) = f_n$
$ls(p_i) = f_{(i-1)} \quad (i = 2 \ldots n)$
$rs(p_i) = f_i \quad (i = 1 \ldots n),$

additional variables $X = \{x : philo; y, z : fork\}, type(THINK) = type(EAT) = philo, \quad type(FORK) = fork$ and initial $\underline{\underline{diphi}}$-algebra A with $A_{philo} = \{P_1, \ldots, P_n\}$ and $A_{fork} = \{F_1, \ldots, F_n\}$. The usual initial marking for the dining philosophers is to have all philosophers $P_1 \ldots P_n$ on place THINK and all forks $F_1 \ldots F_n$ on place FORK, i.e.

$$init = \sum_{i=1}^{n}(P_i, THINK) \oplus \sum_{i=1}^{n}(F_i, FORK).$$

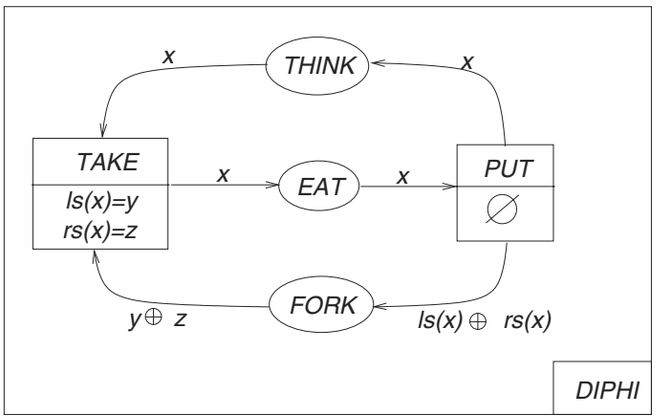
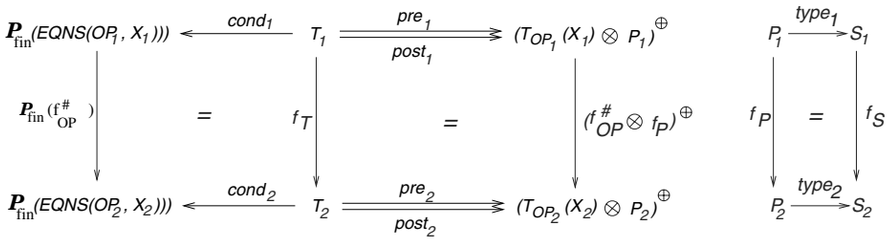


Fig. 1. AHL Net for Dining Philosophers

◊

Definition 2.4 (Category of AHL-Nets).
The category **AHL-net** has AHL-nets N as objects and AHL-net-morphisms $f : N_1 \to N_2$ as arrows where $f = (f_{SPEC}, f_P, f_T, f_A)$ with

- $f_{SPEC} : SPEC_1 \to SPEC_2$ is a specification morphism with sort part $f_s : S_1 \to S_2$, signature part $f_{OP} : (S_1, OP_1, X_1) \to (S_2, OP_2, X_2)$ and extension $f^{\sharp}_{OP}$ to terms and equations such that the restrictions $f_{OP}|_{Var(t_1)} : Var(t_1) \to Var(f_T(t_1))$ of f_{OP} to variables of transitions are bijective;
- $f_P : P_1 \to P_2 \quad and \quad f_T : T_1 \to T_2$ are functions;
- $f_A : A_1 \to A_2$ is induced by an isomorphism $f_a : A_1 \xrightarrow{\sim} V_{f_{SPEC}}(A_2)$ in the category of $SPEC_1$-algebras (see [EM85]) requiring that the following diagrams commute separately for pre- and post-functions.



△

Remark Although this definition is slightly different from those in [PER95] and [EGP99] we still have the existence of pushouts and other HLR-properties as stated in the mentioned papers. These properties allow to define the union of nets via pushouts and net transformations via double pushouts in the sense of high-level replacement systems (see [EGP99]). The bijectivity of f_{OP} on variables of transitions is used in Fact 3.2.2.

Definition 2.5 (AHL-Occurrence Net).
A (deterministic) *AHL-occurrence net* K is an AHL-net

$$K = (SPEC, P, T, pre, post, cond, type, A)$$

such that for all $t \in T$ with $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ and notation $\bullet t = \{p_1, \ldots, p_n\}$ and similarly $t\bullet$ we have

1. (*Unarity*): $\bullet t, t\bullet$ are sets rather than multisets for all $t \in T$, i.e., for $\bullet t$ the places $p_1 \ldots p_n$ are pairwise distinct. Hence $|\bullet t| = n$ and the arc from p_i to t has a unary arc-inscription $term_i$ (rather than a proper sum of terms as in Remark 1 of Definition 2.1).
2. (*No Forward Conflicts*): $\bullet t \cap \bullet t' = \emptyset$ for all $t, t' \in T, t \neq t'$
3. (*No Backward Conflicts*): $t\bullet \cap t'\bullet = \emptyset$ for all $t, t' \in T, t \neq t'$
4. (*Partial Order*): the causal relation $< \subseteq (P \times T) \cup (T \times P)$ defined by the transitive closure of

   $$\{(p,t) \in P \times T \mid p \in \bullet t\} \cup \{(t,p) \in T \times P \mid p \in t\bullet\}$$

   is a finitary strict partial order, i.e. the partial order is irreflexive and for each element in the partial order the set of its predecessors is finite.

△

Remarks

1. Conditions 1-4 are exactly those for occurrence nets in the case of – low-level place/transition nets with $pre(t) = \sum_{i=1}^{n} p_i$ (see [MM90]) and do not take into account possible assignment conflicts formally introduced in Definition 3.7). This means that the flattening $Flat(K)$ of an AHL-occurrence net K may not be a low-level occurrence net (see Example 3.5.2).
2. If we drop the unarity condition, K is called an *AHL- multi-occurrence net.*
3. We only define deterministic AHL-occurrence nets leading to deterministic AHL-processes. The nondeterministic case can be obtained by dropping the second condition "no forward conflicts" because conflicts are allowed in nondeterministic processes.

Definition 2.6 (AHL-process).
A *(deterministic) AHL-process* of an AHL-net N is an AHL-net morphism $p : K \to N$, where K is a (deterministic) AHL-occurrence net with the same data type part $(SPEC, A)$, which is preserved by p, i.e. $p_{SPEC} = id_{SPEC}$ and $p_A = id_A$.

△

Remarks

1. This definition generalizes that of (deterministic) processes of Place/Transition nets which is obtained in the special case where $SPEC$ consists of one sort and A of one element (black token) only.
2. If K is an AHL-multi-occurrence net, then $p : K \to N$ is called *AHL-multi-process*. *AHL-multi-process* seem to be an interesting alternative to AHL-processes in the high-level case as shown in the following examples.
3. We only define deterministic AHL-processes, but nondeterministic processes are obtained if K is a nondeterministic AHL-occurrence net (see Remark 3 of Definition 2.5).

Example 2.7 (AHL-processes of Dining Philosophers).
The AHL-net $COURSE_1$ resp. $COURSE_2$ in Figure 2 with the same data type part as that of $DIPHI$ in Example 2.3 is a (deterministic) AHL-occurrence net resp. AHL-multi-occurrence net, because $COURSE_2$ violates the unitarity condition (see Definiton 2.5.1). Moreover, we obtain a (deterministic) AHL- process $p_1 : COURSE_1 \to DIPHI$ resp. (deterministic) AHL- multi-process $p_2 : COURSE_2 \to DIPHI$ by defining the morphisms p_1 and p_2 as suggested by the labelling in Figure 2:

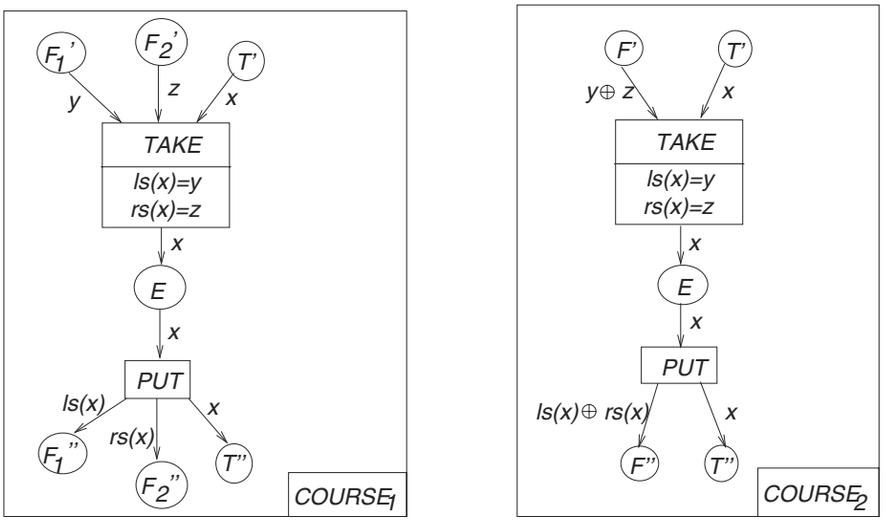


Fig. 2. AHL-process and AHL-multi process of AHL-net $DIPHI$ with AHL-occurrence net $COURSE_1$ and AHL-multi occurrence net $COURSE_2$

More precisely, $p_1 : COURSE_1 \to DIPHI$ maps F_1', F_2', F_1'', F_2'' to FORK, T' and T'' to THINK and E to EAT in $DIPHI$. Similarly $p_2 : COURSE_2 \to$

$DIPHI$ maps F', F'' to FORK, T', T'' to THINK, and E to EAT, while transitions are mapped identically in both cases.

The mappings $p_1 : COURSE_1 \to DIPHI$ and $p_2 : COURSE_2 \to DIPHI$ are both AHL-net morphisms preserving the data type parts. We only show compatibility of p_1 for TAKE with pre-domains:

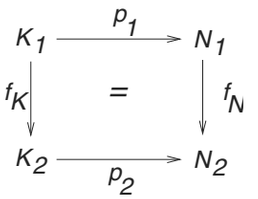
$$\begin{aligned}(id \otimes p_{1_P})^{\oplus}(pre_1(TAKE)) &= (id \otimes p_{1_P})^{\oplus}((y, F_1') \oplus (z, F_2') \oplus (x, T')) \\ &= (y, FORK) \oplus (z, FORK) \oplus (x, THINK) \\ &= pre_{DIPHI}(TAKE) \\ &= pre_{DIPHI}(p_{1_T}(TAKE))\end{aligned}$$

It is interesting to note that the two processes of $DIPHI$ in Figure 2 have different behaviors as formally shown for their flattenings in Section 3.

Finally let us note that we have an AHL-net morphism $f : COURSE_1 \to COURSE_2$ which maps F_1' and F_2' to F' and F_1'' and F_2'' to F''. This leads to an AHL-multi-process morphism (f, id_{DIPHI}) from p_1 to p_2 (see below).

◊

Definition 2.8 (Category of AHL-processes).
Given AHL-processes $p_1 : K_1 \to N_1$ and $p_2 : K_2 \to N_2$ an AHL-process morphism is a pair (f_K, f_N) of AHL-net morphisms such that the following diagram commutes:



The category **AHL-Proc** has AHL-processes as objects and AHL-process morphisms as arrows. Similarly **AHL-Multi-Proc** is the category consisting of AHL-multi-processes as objects and AHL-process morphisms between AHL-multi-processes as morphisms. △

AHL-Proc and **AHL-Multi-Proc** are well-defined because they form diagram categories over **AHL-Net** (see Definition 2.4).

Remark In the case of low-level nets there is also another view how to define a category of processes: Inspired by the "arrows as computations" philosophy, in [DMM89] a category of processes is defined where processes are viewed as arrows of a category in which objects are the possible states of the net (markings) and the source and target of a process are the starting and final state for the process itself.

3 Flattening of AHL-processes

In this section we review the well-known flattening construction from AHL-nets to Place/Transition nets and apply it to construct the flattening of AHL-processes introduced in the previous section. We will see that the flattening of an AHL-process is in general not a low- level process of place/transition nets, because the flattening may contain forward and/or backward conflicts. We call a place of an AHL-occurrence net K to be in forward resp. backward assignment conflict if there is a corresponding place in $Flat(K)$ which is in forward resp. backward conflict. The main result of this section is a characterization of assignment conflicts for AHL-processes. Moreover, we discuss the relationship between the AHL-process of an AHL-net N and the low-level processes of the flattening $Flat(N)$ of N.

Definition 3.1 (Place/Transition Nets and Processes).
A *Place/Transition net* (*P/T net*),

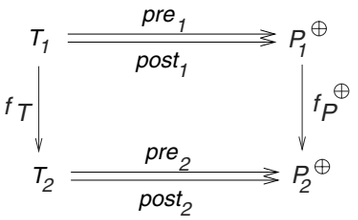
$$N = (P, T, pre, post)$$

consists of sets P and T of places and transitions respectively, and pre- and post-domain functions

$$pre, post : T \to P^{\oplus}$$

where $P^{\oplus}$ is - as in Definition 2.1 - the free commutative monoid over P.

A *P/T net morphism* $f : N_1 \to N_2$ is given by $f = (f_P, f_T)$ with functions $f_P : P_1 \to P_2$ and $f_T : T_1 \to T_2$ such that the following diagram commutes separately for pre- and post-functions:



△

The category **Net** consists of P/T nets as objects and P/T-net morphisms as morphisms.

A *(deterministic) P/T process* of a P/T net N is a P/T net morphism $p : K \to N$, where K is a (deterministic) occurrence net, i.e. a net satisfying conditions 1-4 of Definition 2.5.

Remark The case of nondeterministic P/T processes can be obtained by dropping condition 2 which requires the absence of no forward conflicts. A slightly more restrictive notion of nondeterministic processes is considered in [Eng91]. If condition 1 is dropped, K is called multi-occurrence net and $p : K \to N$ a P/T multi-process of N.

Definition 3.2 (Flattening).

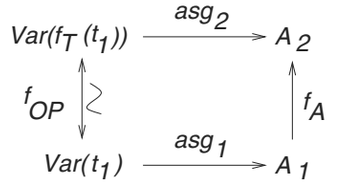
1. Given an AHL-net $N = (SPEC, P, T, pre, post, cond, type, A)$, the flattening *Flat(N) of N* is a P/T net

$$Flat(N) = (CP, CT, pre_A, post_A)$$

defined by
   - $CP = A \otimes P = \{(a, p) | a \in A_{type(p)}, p \in P\}$, called *colored places*;
   - $CT = \{(t, asg) | t \in T, asg : Var(t) \to A$, s.t. all equations $e \in cond(t)$ are valid in A under $asg\}$, called *consistent transition assignments*;
   - $pre_A(t, asg) = \sum_{i=1}^{n} (\overline{asg}(term_i), p_i)$ for $pre(t) = \sum_{i=1}^{n} (term_i, p_i)$ where $term_i \in T_{OP}(X)_{type(p_i)}$ and $\overline{asg} : T_{OP}(Var(t)) \to A$ is the extension of the assignment $asg : Var(t) \to A$ to terms (see [EM85]);
   - $post_A(t, asg) = \sum_{i=1}^{m} (\overline{asg}(term'_i), p'_i)$ for $post(t) = \sum_{i=1}^{m} (term'_i, p'_i)$.
2. Given an AHL-net morphism $f : N_1 \to N_2$ with $f = (f_{SPEC}, f_P, f_T, f_A,)$, the *flattening Flat(f) of f* is a P/T-net morphism given by

$$Flat(f) = (f_A \otimes f_P : CP_1 \to CP_2, f_C : CT_1 \to CT_2)$$

where f_C is defined by $f_C(t_1, asg_1) = (f_T(t_1), asg_2)$ with asg_2 given by



(bijectivity of f_{OP} on variables of transitions is required in Definition 2.3).

△

Fact 3.3 (Flattening Functor).

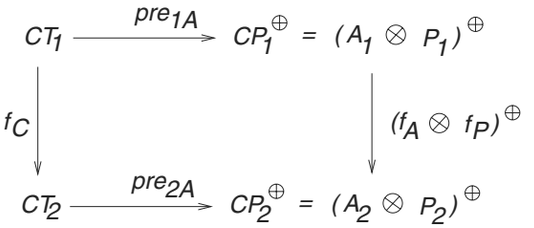
The flattening conctruction defined in Definition 3.2 is well-defined and can be turn into a functor

$$Flat : \textbf{AHL-net} \to \textbf{Net}$$

△

Proof

1. Flat (N) is a well-defined P/T net if we show $pre_A(t, asg), post_A(t, asg) \in CP^{\oplus} = (A \otimes P)^{\oplus}$. In fact we have $\overline{asg}(term_i) \in A_{type(p_i)}$ because $term_i \in T_{OP}(X)_{type(p_i)}$
   by definition of $pre(t) \in (T_{OP}(X) \otimes P)^{\oplus}$. This implies $pre_A(t, asg) = \sum_{i=1}^{n}(\overline{asg}(term_i), p_i) \in (A \otimes P)^{\oplus}$ and similarly for $post_A(t, asg)$.
2. For symmetry reasons it suffices to show commutativity of



   Given $(t_1, asg_1) \in CT_1$ we have for $pre_1(t_1) = \sum_{i=1}^{n}(term_i, p_i)$

$$(f_A \otimes f_P)^{\oplus}(pre_{1A}(t_1, asg_1)) = (f_A \otimes f_P)^{\oplus}(\sum_{i=1}^{n}(\overline{asg_1}(asg_1)(term_i), p_i)$$
$$= \sum_{i=1}^{n}(f_A(\overline{asg_1}(term_i)), f_P(p_i)) \quad (1)$$

$$(pre_{2A}(f_C(t_1, asg_1)) = (pre_{2A}(f_T(t_1), asg_2)$$
$$= \sum_{i=1}^{n}(\overline{asg_2}(f_{OP}^{\#}(term_i)), f_P(p_i)) \quad (2)$$

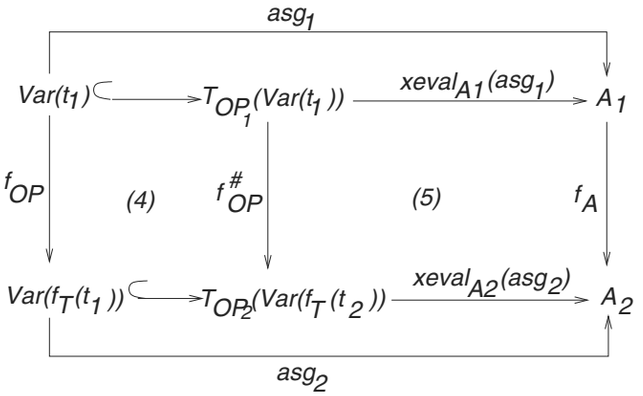
   where the last equation holds, because f AHL-net morphism implies:

$$pre_2(f_T(t_1)) = (f_{OP}^{\#} \otimes f_P)^{\oplus}(\sum_{i=1}^{n}(term_i, (p_i))$$
$$= (f_{OP}^{\#} \otimes f_P)^{\oplus}(\sum_{i=1}^{n}(term_i, p_i))$$
$$= \sum_{i=1}^{n}(f_{OP}^{\#}(term_i), f_P(p_i)).$$

   Comparing (1) and (2) is suffices to show

$$\overline{asg_2}(f_{OP}^{\#}(term_i)) = f_A(\overline{asg_1}(term_i)) \quad (i = 1 \ldots n). \quad (3)$$

For (3) it is sufficient to show commutativity of (5) in the following diagram:



But due to commutativity of (4) and the outer diagram by definition of asg_2 diagram (5) commutes because $T_{OP_1}(Var(t_1))$ is a free construction (see [EM85]).

3. Obviously we have $Flat(f) = id_{Flat(N)}$ for $f = id_N$. We can show $Flat(g \circ f) = Flat(g) \circ Flat(f)$ by composing the corresponding diagrams in part 2 of the proof and using the properties
$(g \circ f)_C = g_C \circ f_C$ and $((g \circ f)_A \otimes (g \circ f)_P)^{\oplus} = (g_A \otimes g_P)^{\oplus} \circ (f_A \otimes f_P)^{\oplus}$.

Example 3.4 (Flattening of Dining Philosophers).
Given the AHL-net $DIPHI$ for dining philosophers in Example 2.3 we obtain the following flattening

$$Flat(DIPHI) = (CP, CT, pre_A, post_A)$$

with

- $CP = A \otimes P = \{(P_i, T), (P_i, E), (F_i, F) | i = 1 \dots n\}$
- $CT = \{(TAKE, asg_i), (PUT, asg_i) | asg_i(x) = P_i, i = 1 \dots n\}$
  where $T = THINK, E = EAT, F = FORK$ and $asg_i(y), asg_i(z)$ are uniquely determined by $asg_i(x) = P_i$ and the equations $ls_A(x) = y$ and $rs_A(x) = z$. The net $Flat(DIPHI)$ in the case $n = 3$ is shown in Figure 3:

◊

Remark 3.5 (Flattening of AHL-processes).
As we will show in the following examples, an AHL- process $p : K \to N$ can be flattened to

$$Flat(p) : Flat(K) \to Flat(N),$$

but $Flat(p)$ is not necessarily a P/T process of $Flat(N)$. In the first example $Flat(p)$ corresponds to a set of P/T-processes and in the second one $Flat(K)$ is

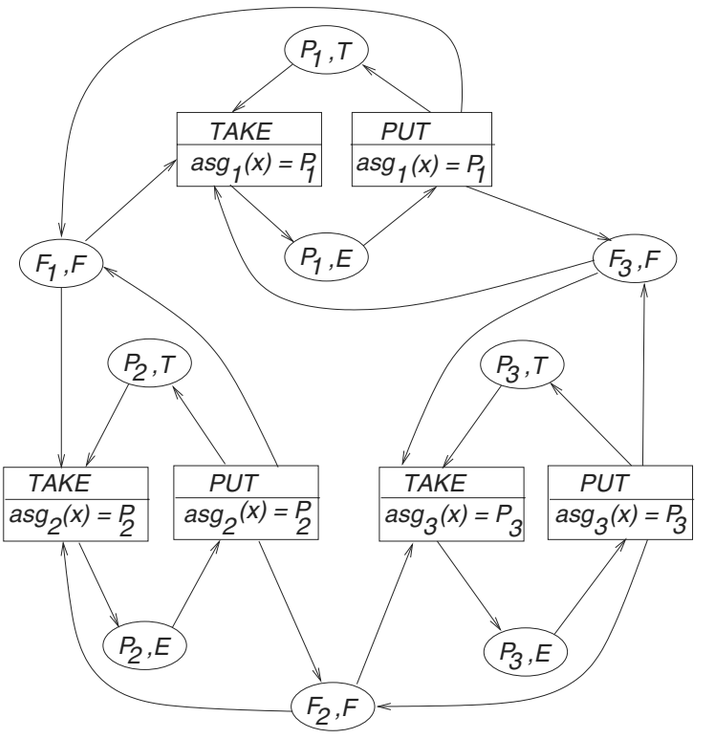


Fig. 3. Flattening of Dining Philosophers ($n = 3$)

not a low-level occurrence net, although K is a high-level occurrence net. This is due to the fact that K has assignment conflicts which become conflicts in $Flat(K)$. In Definition 3.7 we will formally define assignment conflicts and in Fact 3.8 we will give a characterization of them.

◊

Example 3.6 (Flattening of AHL-processes).

1. The AHL-process $p_1 : COURSE_1 \to DIPHI$ (see 2.7) has the flattening

$$Flat(p_1) : Flat(COURSE_1) \to Flat(DIPHI)$$

where $Flat(DIPHI)$ is given in example 3.4, $Flat(COURSE_1)$ in Figure 4 and $Flat(p_1)$ maps places (P_i, T') and (P_i, T'') to (P_i, T), (P_i, E) to (P_i, E), and (F_i, F'_j), (F_i, F''_j) to (F_i, F) for $i \in \{1, 2, 3\}, j \in \{1, 2\}$.
Obviously $Flat(p_1)$ corresponds to three processes of $Flat(DIPHI)$ with usual initial marking $init = \sum_{i=1}^{3}(P_i, T) \oplus (F_i, F)$. Since we have not considered initial markings up to now, $Flat(p_1)$ is formally a (single) process

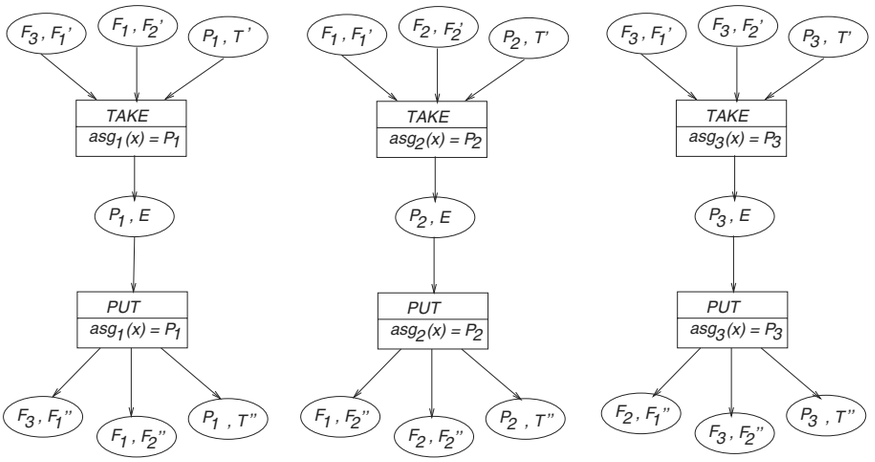


Fig. 4. Flattening of AHL-process $p_1 : COURSE_1 \rightarrow DIPHI$

of $Flat(DIPHI)$. The usual interpretation of $Flat(p_1)$ would have as initial marking one token on each input place, which would be mapped to the marking

$$init_2 = \sum_{i=1}^{3}(P_i, T) \oplus 2(F_i, F)$$

where two forks are on each fork place. This would avoid conflicts, but does not correspond to the problem of shared resources. Taking into account initial markings (see Subsection 4.4), $Flat(p_1)$ is not a process of $(Flat(DIPHI), init)$, but only a set of three single processes.

2. The flattening of the AHL-multi-process $p_2 : COURSE_2 \rightarrow DIPHI$ (see Example 2.7) leads to the following flattening:

$$Flat(p_2) : Flat(COURSE_2) \rightarrow Flat(DIPHI)$$

where $Flat(DIPHI)$ is given in Example 3.4, $Flat(COURSE_2)$ in Figure 5 and $Flat(p_2)$ maps places (P_i, T') and (P_i, T'') to (P_i, T'), (P_i, E) to (P_i, E), and (F_i, F'), (F_i, F'') to (F_i, F) for $i \in \{1, 2, 3\}$.

The P/T net $Flat(COURSE_2)$ is neither an occurrence net, nor a multi-occurrence net, although $COURSE_2$ is an AHL-multi-occurrence net. Especially $COURSE_2$ has no forward and backward conflicts, but $Flat(COURSE_2)$ has three forward conflicts for places (F_i, F') and 3 backward conflicts for places $(F_i, F''), i \in \{1, 2, 3\}$.

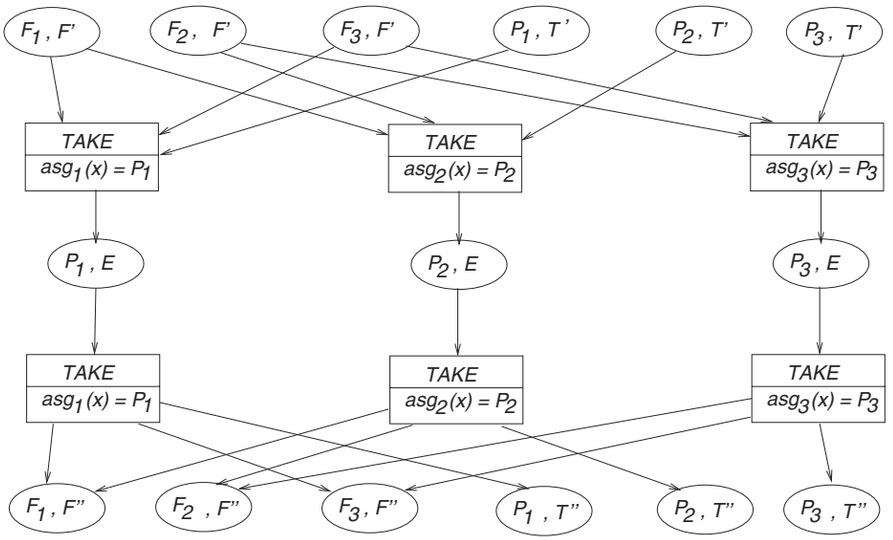


Fig. 5. Flattening of AHL-multi-process $p_2 : COURSE_2 \to DIPHI$

In fact we have in $COURSE_2$

$$pre(TAKE) = (x, T') \oplus (y, F') \oplus (z, F').$$

This implies for asg_1 with $asg_1(x) = P_1, \quad asg_1(y) = F_3, \quad asg_1(z) = F_1$:

$$pre_A(TAKE, asg_1) = (P_1, T') \oplus (F_3, F') \oplus (F_1, F').$$

But (F_1, F') is also in the pre-domain of $(TAKE, asg_2)$ leading to a forward conflict in $Flat(COURSE_2)$.
According to the following definition of assignment conflicts (see Definition 3.7) the place F' of $COURSE'_2$ is in forward assignment conflict, because (F_1, F') defines a forward conflict in the flattening $Flat(COURSE_2)$ of $COURSE_2$.

3. If we drop the equations of transition $TAKE$ of $COURSE_1$ and $DIPHI$, then $COURSE'_1$ is an AHL-process of $DIPHI'$ and we have two different consistent assignments asg_1, asg'_1 with $asg_1(x) = P_1, asg_1(y) = F_3, asg_1(z) = F_1$ and $asg'_1(x) = P_1, asg'_1(y) = F_1, asg'_1(z) = F_3$ leading to a forward assignment conflict for place T', because (P_1, T') is in the pre-domain of the transitions $(TAKE, asg_1)$ and $(TAKE, asg'_1)$. Note that asg_1 is not a consistent assignment for $COURSE_1$ and $DIPHI$, such that we have no transition $(TAKE, asg'_1)$ in $Flat(COURSE_1)$ and $Flat(DIPHI)$.
This example shows that also AHL-processes may have assignment conflicts, while Example 2 has shown assignment conflicts for the AHL-multi-process $p_2 : COURSE_2 \to DIPHI$.

$\Diamond$

Definition 3.7 (Assignment Conflicts).
Given a (deterministic) AHL-occurrence net

$$K = (SPEC, P, T, pre, post, cond, type, A)$$

a place $p \in P$ is in *forward* (resp. *backward) assignment conflict* in K if there is a data $a \in A_{type(p)}$ such that the place $(a, p) \in A \otimes P$ defines a forward (resp. backward) conflict in the flattening $Flat(K)$ of K.

△

Remark According to definition 2.5 $(a, p) \in A \otimes P$ defines a forward (resp. backward) conflict in $Flat(K)$ if there are distinct transitions $(t, asg), (t', asg') \in CT$ with

$$(a, p) \in [pre_A(t, asg)] \cap [pre_A(t', asg')]$$

(forward conflict) resp.

$$(a, p) \in [post_A(t, asg)] \cap [post_A(t', asg')]$$

(backward conflict).

In the following we give a characterization for forward and backward assignment conflicts:

Theorem 3.8 (Characterization of Assignment Conflicts).
Given a (deterministic) AHL-occurrence net K as in 3.7 a place $p \in P$ is in forward (resp. backward) assignment conflict in K if and only if the following *assignment conflict condition* is satisfied:

There is a transition $t \in T$ with $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ (resp. $post(t) = \sum_{i=1}^{n}(term_i, p_i)$) such that $p = p_i$ for some $i \in \{1, \ldots, n\}$ and there are consistent assignments $asg \neq asg'$ with

$$\overline{asg}(term) = \overline{asg'}(term) = a$$

for some $a \in A_{type(p)}$ and $term = term_i$ as shown in Figure 6.

△

Remark In the case of AHL-multi-occurrence nets K the assignment condition is slightly more general because we require the existence of $t \in T$ with $p = p_i = p_j$ for some
$i, j \in \{1, \ldots, n\}$ and consistent assignments $asg \neq asg'$ with

$$\overline{asg}(term_i) = \overline{asg'}(term_j) = a \in A_{type(p)}$$

which includes the condition above for $i = j$.

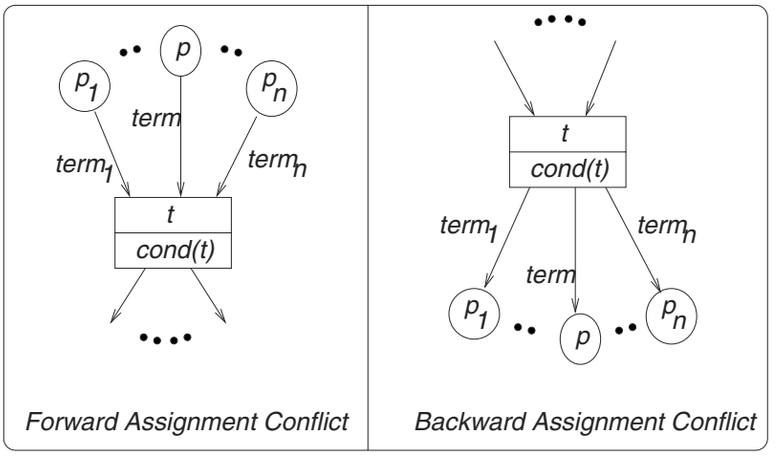


Fig. 6. Forward and Backward Assignment Conflicts with $(t, asg) \neq (t, asg') \in CT$ and $\overline{asg}(term) = \overline{asg'}(term)$

Proof For symmetry reasons it suffices to consider forward conflicts.

Necessity of Assignment Conflict Condition

Given a place $p \in P$ in forward assignment conflict we have $(a, p) \in A \otimes P, (t, asg) \neq (t', asg') \in CT$ with

$$(a, p) \in [pre_A(t, asg)] \cap [pre_A(t', asg')].$$

Let $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ and $pre(t') = \sum_{j=1}^{m}(term'_j, p'_j)$ then

$$pre_A(t, asg) = \sum_{i=1}^{n}(\overline{asg}(term_i), p_i), pre_A(t', asg') = \sum_{j=1}^{m}(\overline{asg'}(term'_j), p'_j)$$

which means that we have $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$ with

$$(a, p) = (\overline{asg}(term_i), p_i) = (\overline{asg'}(term'_j), p'_j).$$

This implies $p = p_i = p'_j$ and $a = \overline{asg}(term_i) = (\overline{asg'}(term'_j)$.

Case 1 ($t \neq t'$) In this case we have $p = p_i = p'_j \in \bullet t \cap \bullet t'$ for $t \neq t'$ and hence a forward conflict in K, which contradicts the fact that K is an AHL-occurrence net.

Case 2 ($t = t'$) In this case $p = p_i = p'_j$ and $pre(t) = pre(t')$ implies $term_i = term'_j$ such that we have for $term = term_i$ the assignment conflict condition

$$\overline{asg}(term) = \overline{asg'}(term) = a$$

with $asg \neq asg'$, because $(t, asg) \neq (t', asg')$ and $t = t'$.

Sufficiency of Assignment Conflict Condition:
Given the assignment conflict condition we have $asg \neq asg'$ with

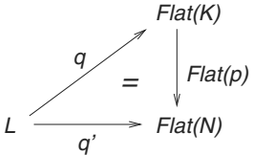
$$(a, p) \in [pre_A(t, asg)] \cap [pre_A(t', asg')].$$

This implies $(t, asg) \neq (t, asg')$ and hence (a, p) defines a forward conflict in $Flat(K)$, i.e. a forward assignment conflict for p in K.

Remark In the case of an AHL-multi-occurrence net K case 2 above allows $term_i \neq term_j$ for $p_i = p'_j$ and hence the slightly more general assignment conflict condition.

Remark 3.9 (Relationship between High-Level and Low-Level Processes).

1. Given an AHL-process $p : K \to N$ we have seen already that
   $Flat(p) : Flat(K) \to Flat(N)$ is not necessarily a low-level process of $Flat(N)$ because K may have an assignment conflict which causes a conflict in $Flat(K)$. But if K has no assignment conflicts, then $Flat(K)$ has no conflicts and $Flat(p)$ is a low-level process. Moreover, even in the general case with assignment conflicts of K each low-level process $q : L \to Flat(K)$ defines a low-level process $q' = Flat(p) \circ q : L \to Flat(N)$ of $Flat(N)$. In this way we obtain the set of all *low-level* processes of $Flat(N)$ *associated* to $p : K \to N$.
2. Vice versa let us call a low-level process $q' : L \to Flat(N)$ *high-level-representable*, if there is an AHL-process $p : K \to N$ and a P/T-process $q : L \to Flat(K)$ such that $q' = Flat(p) \circ q : L \to Flat(N)$. For example let L_1, L_2, L_3 be the three connected low-level occurrence nets in Figure 4 leading to low-level processes $q'_i = Flat(DIPHI) \quad (i = 1 \ldots 3)$. Then each of these processes is high-level representable by the AHL-process $p_1 : COURSE_1 \to DIPHI$, where
   $q'_i : L_i \to Flat(COURSE_1)$ is the embedding of L_i into $Flat(COURSE_1)$ in Figure 4.
   In general, we claim that (under mild assumptions which we dont know yet) each low-level process $q' : L \to Flat(N)$ is high-level representable by some AHL-processes $p : K \to N$, where the net structures of K and L coincide, i.e. the skeleton of K is isomorphic to L. In other words, $q' : L \to Flat(N)$ can be lifted to a high-level process $p : K \to N$ with some $q : L \to Flat(K)$.



◊

4 Future Perspectives

In this section, we discuss several extensions of AHL-processes and flattening introduced in the previous sections. Each of these extensions can be considered in its own right, but most of them can be also combined with each other. A more detailed presentation of most of these extensions will be given in forthcoming papers.

4.1 High-Level Net Processes

We have defined processes for AHL-nets in Section 2, where AHL-nets, however, are only one specific type of high-level nets. Of course the question arises how far this can be generalized to other types of high-level nets, like colored nets or predicate transition nets. In [EP97a, EP97c, PE01] we have introduced the notion of parameterized net classes, which can be instantiated to a great variety of low-level and high-level net classes in the literature including colored nets [Jen92] and predicate transition nets [Gen91]. The categorical theory of parameterized net classes is based on the notion of abstract Petri nets introduced in [Pad96]. This notion relies on an institution for the data type part and on a net structure functor $Net : \mathbf{Sets} \to \mathbf{Sets}$ for the net structure part. In [Pad96] the instantiation for P/T nets is given by $Net(P) = P^{\oplus}$ and for AHL-nets by $Net(P) = (T_{OP}(X) \times P)^{\oplus}$. The basic concepts of a theory of abstract Petri nets are developed, including the category of abstract Petri nets and a flattening construction from high-level to low-level abstract Petri nets. But a concept of processes for abstract Petri nets generalizing processes for elementary nets and P/T nets is not yet considered in [Pad96] or elsewhere. As discussed in Section 2 the main problem is to find a suitable notion of occurrence net in the case of high-level nets. In fact, our notion of AHL-occurrence net and process in Definitions 2.5 and 2.6 can be generalized to abstract Petri nets, if we are able to define the pre-domain $\bullet t$ and the post-domain $t\bullet$ of a transition $t \in T$. The present version of abstract Petri nets in [Pad96] does not allow to define these domains as subsets of the set of places P. For this purpose an idea could be to extend the notion of abstract Petri nets by a natural transformation

$$[\quad] : Net \to \mathcal{P}$$

from the net-structure functor $Net : \mathbf{Sets} \to \mathbf{Sets}$ to the power set functor $\mathcal{P} : \mathbf{Sets} \to \mathbf{Sets}$, where the family of functions

$$[\quad]_P : Net(P) \to \mathcal{P}(P)$$

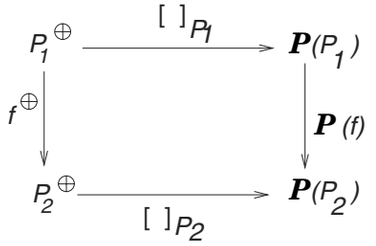
is intended to define for each marking $m \in Net(P)$ over a set P of places the subset $[m]_P \subseteq P$ of places occurring in the marking m. In the case of P/T nets we have $Net(P) = P^{\oplus}$ and $[]_P : P^{\oplus} \to \mathcal{P}(P)$ can be defined for $m = \sum_{i=1}^{n}(\lambda_i, p_i)$ with $p_i \in P$ and $\lambda_i \in \mathbb{N} - \{0\}$ by

$$[m]_P = \{p_1, \ldots p_n\} \subseteq P.$$

Similarly in the case of AHL-nets with $Net(P) = (T_{OP}(x) \times P)^{\oplus}$ and $m = \sum_{i=1}^{n}(term_i, p_i)$ with $p_i \in P$ and $term_i \in T_{OP}(X)$ by

$$[m]_P = \{p_1, \ldots p_n\} \subseteq P.$$

In both cases we obtain a natural transformation, which means in the case of P/T nets commutativity of the following diagram for each function $f : P_1 \to P_2$.



In fact, we have for $m = \sum_{i=1}^{n}(\lambda_i, p_i)$

$$\begin{aligned}\mathcal{P}(f)([m]_{P_1}) &= \mathcal{P}(f)\{p_1, \ldots p_m\} = \{f(p_1), \ldots f(p_m)\} \\ &= [\sum_{i=1}^{n}(\lambda_i, f(p_i))]_{P_2} = [f^{\oplus}(m)]_{P_2}.\end{aligned}$$

Using this natural transformation we are able to define pre-domain $\bullet t$ and post-domain $t\bullet$ of a transition $t \in T$ by

$$\bullet t = [pre(t)]_P \subseteq P \quad and \quad t\bullet = [post(t)]_P \subseteq P$$

With these notions we can generalize at least the conflict- and partial order conditions in Definition 2.5 to abstract Petri nets. In order to define unarity we must be able to distinguish between set and multisets which would require an additional extension of abstract Petri nets.

Finally let us note that AHL-nets as defined in Definition 2.1 of this paper are slightly different from those in [PER95] and [Pad96]. In our case we have

$$Net(P) = (T_{OP}(X) \times P)^{\oplus},$$

where $T_{OP}(X) \otimes P \subseteq T_{OP}(X) \times P$ depends on the type function $type : P \to S$, which is not present in the definition of AHL-nets in [PER95] and [Pad96]. In fact, our definition is much more adequate in the case of flattening: Otherwise in our running example (and similar for most other examples) the places of the flattened net would include also pairs $(P_i, FORK)$ and $(F_i, THINK)$, where philosophers P_i are on the place $FORK$ or forks F_i on the place $THINK$. On the other hand Net in our case is no longer a functor $Net : \mathbf{Sets} \to \mathbf{Sets}$, but a slightly different functor $Net : (\mathbf{Sets} \downarrow S) \to \mathbf{Sets}$, because $type : P \to S$ is an element of the comma category $\mathbf{Sets} \downarrow S$.

This, however, would require an extension of the notion of abstract Petri nets in [Pad96].

4.2 Higher Order Net Processes

In [Hof00, Hof01] we have introduced the concept and formal definition of Algebraic Higher Order Nets to model flexible business processes. Algebraic Higher Order Nets are AHL-nets where the data type part is extended by higher order types, sorts and functions. This allows to have functions as data items on places which are typed by higher order sorts, such that different functions may be activated and applied during run time. This feature is especially useful to model flexible business processes, where the choice of different functions would require different nets, in the case of ordinary AHL-nets, but can be modeled with a single Algebraic Higher Order Net using different functions as tokens. For a restricted class of Algebraic Higher Order Nets it is possible to define an unfolding operation leading to an AHL-net. This is similar in some sense to the flattening construction of AHL-nets leading to Place/Transition nets discussed in Section 3. But in general Algebraic Higher Order Nets are more expressive than AHL-nets.

AHL-nets in contrast to Algebraic Higher Order Nets are adequate in application domains, where the context is known from the very beginning. The system can be modeled by a Petri net with a fixed net structure, that is, changes of the environment can only be modeled by changing the structure of Petri nets. But in other application domains like business processes it is also desirable to support the fact that an organisation of a system is not fixed once and for all. Rather, a large variety of changes e.g. replacing one task by another refined task can occur. In such application domains it is useful to use Algebraic Higher Order Nets. In [Hof01] we have used Algebraic Higher Order Nets in an example of logistic processes and discussed different kinds of changes which are supported by this formalism. In a forthcoming paper this example will be extended to a whole case study where specific scenarios will be modeled as processes for AHL-nets and Algebraic Higher Order Nets.

In fact, the main difference between Algebraic Higher Order Nets and AHL-nets consist in the use of higher order functions in the data type part. This means that the extension of high-level occurrence nets and processes to higher order occurrence nets and higher order processes seems to be quite natural and will be studied as part of ongoing work.

4.3 Open AHL-processes

In [BCEH01] we have formally defined the notion of open P/T net and process of such P/T nets leading to a compositional modeling of reactive systems. The main result in [BCEH01] is an amalgamation theorem which allows the amalgamation and decomposition of open P/T nets. This kind of result is not possible for ordinary P/T nets, because ordinary P/T nets cannot take into account the

behavior of the environment. On the other hand this is the essential new feature of open nets, where in addition to producing and consuming tokens from places by firing of transitions we have open places, where so called "invisible actions" can produce or consume tokens on these open places. The intuitive idea is that the open places are the interface towards the environment. The "invisible actions" of one open net component can be interpreted as the consequence of the firing of transitions of other open net components which belong to the environment of the first component sharing the open places as interface places.

This idea of open net has been applied to the modeling of interorganizational workflows in the sense of [vdA98] and on a conceptual level to interoperability in train control systems [PJE+01]. In the second application open high-level processes have been introduced on a conceptual level already, and called scenario nets in [PJE+01]. In this paper we have defined AHL-processes, which can be extended to open nets in the sense of [BCEH01]. In fact, the idea of open places in [BCEH01] allows to define open AHL-nets, open AHL-net morphisms and open AHL-processes of open AHL-nets. As far as we can see it is possible to extend on one hand the theory of open nets and processes of P/T nets in [BCEH01] to open AHL-nets and processes, and on the other hand the flattening of AHL-processes in this paper to open AHL-processes.

4.4 AHL-processes with Initial Marking

For low-level processes $p : K \to N$ it is implicitly assumed that we have an initial marking $init_K$, where each input place of the occurrence net K contains exactly one token. Moreover, if N has an initial marking $init_N$ then it is assumed that $p(init_K) = init_N$. Sometimes it is useful to require $p(init_K) \leq init_N$ or $p(init_K) \geq init_N$.

For high-level processes $p : K \to N$ of an AHL-net with initial marking $init_N$ we propose to have a set of markings $INIT_K$, where each $init_K \in INIT_K$ is a marking of all input places of K, i.e.

$$[init_K] = \{p \in P_K | \neg\exists \quad t \in T_K : \quad p \in [post_K(t)]\},$$

and $p(init_K) = init_N$. Here $[m]$ denotes the set of all places occurring in m i.e. for $m = \sum_{i=1}^{n}(term_i, p_i)$ we have $[m] = \{p_1, \ldots p_n\}$, where unitarity of K implies that $p_1, \ldots p_n$ are pairwise disjoint. The idea to have a set $INIT_K$ of initial markings for the high-level process corresponds to the idea that the process should be defined for different input parameters $init_K \in INIT_K$.

In our running example of dining philosophers we should have n initial markings for the process $p_1 : COURSE_1 \to DIPHI$ (resp. multi-process $p_2 : COURSE_2 \to DIPHI$) if $DIPHI$ has the standard initial marking

$$init_{DIPHI} = \sum_{i=1}^{n}(P_i, THINK) + \sum_{i=1}^{n}(F_i, FORK)$$

In fact, the n initial markings of $COURSE_1$

$$init^{(i)}_{COURSE_1} = (P_i, T_1') \oplus (ls_A(P_i), F_1') \oplus (rs_A(P_i), F_2') \qquad (i = 1 \dots n)$$

correspond to the n cases, where each philosopher $P_i (i = 1 \dots n)$ has his associated forks. For the multi-process we would have

$$init^{(i)}_{COURSE_2} = (P_i, T') \oplus (ls_A(P_i), F') \oplus (rs_A(P_i), F') \quad (i = 1 \dots n),$$

where we have two tokens on F', i.e. the two forks of P_i.

Of course, it would make no sense to have the initial marking

$$init_{COURSE_1} = (P_1, T') \oplus (F_1, F_1') \oplus (F_2, F_2')$$

In this case the transition $TAKE$ could not fire, because the left-side fork of P_1 is F_3 and the right-side fork is F_1. However, it is probably not useful to force by definition that K can fire for each initial marking $init_K \in INIT_K$. But it is interesting to analyze under which condition there is - up to independence of firing - only one firing sequence in K leading from the initial marking $intit_K$ to some final marking fin_K of K, i.e. fin_K contains only tokens on output places of K. For this purpose it makes sense to extend the flattening construction of Section 3 to the case with initial markings:

A marking m in the high-level case of AHL-net N is given by

$$m = \sum_{i=1}^{n} (a_i, p_i) \in (A \otimes P)^{\oplus}.$$

In fact, m can also be interpreted as a marking of the low-level net $Flat(N)$, because $A \otimes P$ is the set of places of $Flat(N)$. Hence for $(N, init)$ we obtain $Flat(N, init)$, where $init$ is initial marking of N and $Flat(N)$.

For a high-level process $p : K \to N$ with set of initial markings $INIT_K$ of K and initial marking $INIT_N$ of N we obtain the following flattenings

$$Flat(p) : Flat(K, init_K) \to Flat(N, init_N) \; for \text{ all } init_K \in INIT_K.$$

As pointed out already $Flat(K)$ is not necessarily a low-level occurrence net. Moreover, $init_K$ is not necessarily a marking of all the input places of $Flat(K)$. For example, $init^{(1)}_{COURSE_1}$ is only a marking for three of the nine input places of $Flat(COURSE_1)$ in Figure 4, where we have $n = 3$ philosophers. But for each initial marking of $COURSE_1$ we obtain one subnet of $Flat(COURSE_1)$ leading to a well-defined low-level process of $Flat(DIPHI, init_{DIPHI})$.

In general, each high-level process $p : K \to N$ with initial markings $INIT_K$ and $init_N$ defines a set of low-level processes of $Flat(N)$ given by

$$LL(p, INIT_K) = \{q' : L \to Flat(N, init_N) | q' = Flat(p) \circ q \text{ for some low-level process } q : L \to Flat(K, init_K) \text{ with } init_K \in INIT_K\}.$$

Vice versa, given a low-level process $q' : L \to Flat(N, init_N)$ we call q' *high-level representable*, if there is an AHL-process $p : K \to N$ with initial markings $INIT_K$ and $init_N$ and a low-level process $q : L \to Flat(K, init_K)$ for some $init_K \in INIT_K$ with $q' = Flat(p) \circ q$. Again the problem arises under which assumptions low-level processes are high- level representable and which sets of low-level processes are realizable by one high-level process.

4.5 AHL-processes with Data Type Behavior

In order to define the data type behavior of AHL-processes it makes sense to consider processes $p : K \to N$ not only with initial markings $INIT_K$ and $init_N$ as discussed above, but also with finite sequences of input and output places (i.e. an arbitrary but fixed order of finite sets instead of arbitrary sets) IN and OUT given by

$$IN = (p_1 \dots p_n) \text{ and } OUT = (p'_1 \dots p'_m).$$

Let $s_i = type(p_i)$ $(i = 1 \dots n)$ and $s_j = type(p'_j)$ $(i = 1 \dots m)$ then we can define input and output parameter sets

$$A = A_{s_1} \times \dots \times A_{s_n} \text{ and } B = A_{s'_1} \times \dots \times A_{s'_m},$$

where A is the data type algebra of the AHL-nets K and N.

The set of all possible final markings of the out-put places of K is given for $OUT = (p'_1 \dots p'_m)$ by

$$FIN_K = \{fin_K \in (A \otimes P)^{\oplus} | \quad \exists b = (a'_1 \dots a'_m) \in B, fin_K = \sum_{j=1}^{m} (a'_j, p'_j)\}$$

In this case let data $(fin_K) = (a'_1 \dots a'_n) = b \in B$. Similarly let $data(init_K) = (a_1 \dots a_n) = a \in A$ for $init_K = \sum_{i=1}^{n} (a_i, p_i)$.

With these preliminaries we can define the *data type behavior* of $(p_i, INIT_K)$ by

$data - type - beh(p, INIT_K)$
$= \{(a, b) \in A \otimes B | \quad \exists init_K \in INIT_K, \exists fin_K \in FIN_K,$ and $\exists$ firing sequence from $init_K$ to fin_K in K with $data(init_K) = a$ and $data(fin_K) = b\}$.

In the case of our process $p_1 : COURSE_1 \to DIPHI$ we have

$$A = A_{fork} \times A_{fork} \times A_{philo} = B$$

and the data type behavior of $(p_1, INIT_{COURSE_1})$ is a partial identity

$data - type - beh(p_1, INIT_{COURSE_1}) : A \multimap\!\!\!\!\circ\!\!\rightarrow B$ defined on the following input parameter values

$$(F_n, F_1, P_1), (F_1, F_2, P_2), \ldots, (F_{n-1}, F_n, P_n).$$

In each case there is exactly one firing sequence from the corresponding initial to the corresponding final marking.

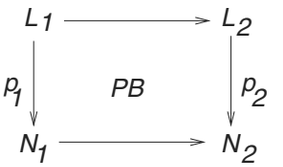
In general it seems to be useful to analyze high-level processes $p : K \to N$ with $(INIT_K, init_N)$ concerning consistency and completeness. Consistency would require that for each $init_K \in INIT_K$ there is at least one (or exactly one up to independence of firing) firing sequence from $init_K$ to some final marking $fin_K \in FIN_K$. Completeness means that $INIT_K$ contains all those markings of the input places $init$, such that there exists a firing sequence to some $fin_K \in FIN_K$.

AHL-nets and processes with data type behavior as discussed above are an important example for the instantiation of the integration paradigm in [EO01a] of integrated data type and process modeling techniques.

4.6 Construction of Low- and High-Level Net Processes

There are several different kinds of process constructions which should be considered for high-level nets:

1. A well-known construction of nondeterministic processes in the low-level case is the *unfolding* of a given net. It is certainly interesting to extend the unfolding constructions and results to open low-level nets in the sense of [BCEH01] on one hand and to high-level nets on the other hand. Especially this is interesting in view of an event structure semantics for these types of Petri nets generalizing the well-known Winskel adjunction of [Win88].
2. The idea of *concatenable processes* [DMM89, MMS97] is another important issue which should be extended to open and high-level nets respectively and also in combination. This allows sequential composition of processes, while parallel composition should be defined via coproducts or tensor products. In the case of open nets both constructions should be a special case of amalgamation in the sense of [BCEH01].
3. Using the constructions above it should be possible to define *process terms* built up from basic processes in analogy to data type terms built up from data type operations (see [EM85]). The closure of a given set of processes w.r.t. process terms would correspond to the construction of all terms and the term algebra respectively. This allows to define *process types* as nets together with a given set of processes for these nets as advocated in [EMP97] already and to study process types in analogy to data types in [EM85].
4. Given a net morphism $f : N_1 \to N_2$, the *translation* of a process $p_1 : L \to N_1$ is given by $p_2 = f \circ p : L \to N_2$ and the *restriction* $p_1 : L_1 \to N_1$ of a process $p_2 : L_2 \to N_2$ can be defined via the following pullback



provided that the corresponding pullback construction exists for the corresponding net class and L_1 becomes an occurrence net. See [BCEH01] for the restriction of open processes for open P/T-nets.

4.7 Component Concept for Low- and High-Level Nets

In [EO01a, EO01b] we have introduced a components concept for integrated data type and process modeling techniques. The basic idea - in analogy to the algebraic module specification concept in [EM90] - is to have for each component an import specification IMP, export specification EXP, body specification BOD and two types of morphisms

$$i : IMP \to BOD \quad and \quad e : EXP \to BOD$$

connecting import and export with the body specification.

An integrated data type and process specification according to [EO01a] is defined on 4 layers: the data type layer 1, the data state and transformation layer 2, the process layer 3, and the system architecture layer 4.

In the case of AHL-nets N layer 1 consists of the algebraic specification $SPEC$ of N, in layer 2 the data states correspond to markings of N and transformations between data states are defined by firing of the transitions of N, and processes in layer 3 should be high-level processes of N as defined in this paper, or open AHL-processes as discussed in 4.1. Up to now, however, there is no component concept for AHL-nets corresponding to layer 4, or to the general component concept in [EO01b]. An instantiation of the general component concept leads to the following ideas of a component concept for AHL-nets.

Roughly spoken import, export and body consist of (open) AHL-nets N_I, N_E, and N_B respectively together with a set of (open) AHL-processes in each case. The morphism $e : EXP \to BOD$ is based on an (open) AHL-net morphism $f_e : N_E \to N_B$ such that the export processes are restrictions (see Subsection 4.7) of corresponding body processes. The morphism $i : IMP \to BOD$ is constructive in the sense of [EO01b]. This means that on one hand the import processes are translated along $i : IMP \to BOD$ to become body processes. On the other hand the body net N_B and the body processes are constructed from corresponding parts of the import and new parts introduced in the body. Especially the body net N_B might be constructed as union (pushout) of N_I and some auxiliary net N_{aux}. In the case of open AHL-nets and processes (see Subsection 4.3) the new body processes might be constructed by amalgamation of import processes of N_I and auxiliary processes of N_{aux}. But also other constructions in the sense of Subsection 4.6 could be used to construct new body processes.

Another alternative - also proposed in Subsection 4.6 and [EO01b] - would be to define the morphism $e : EXP \to BOD$ by a suitable net transformation in the sense of [PER95] and $i : IMP \to BOD$ as net inclusion. This corresponds to the idea that the relationship between export and body is a refinement.

In fact, both alternatives discussed above are not only useful for a component concept of high-level nets, but also for low-level nets, which are also discussed as instantiation of the general integration paradigm in [EO01a, EO01b].

References

[BCEH01] P. Baldan, A. Corradini, H. Ehrig, and R. Heckel. Compositional Modeling of Reactive Systems Using Open Nets. In *Proc. of CONCUR'01*, 2001. To appear.

[DMM89] P. Degano, J. Meseguer, and U. Montanari. Axiomatizing Net Computations and Processes. In *Proc. of LICS'89*, pages 175–185, 1989.

[EGH92] H. Ehrig, M. Große-Rhode, and A. Heise. Specification Techniques for Concurrent and Distributed Systems. Technical Report 92/5, Technical University of Berlin, jan. 1992. Invited paper for 2nd Maghr. Conference on Software Engineering and Artificial Intelligence, Tunis,1992.

[EGP99] 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.

[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.

[EM90] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2: Module Specifications and Constraints*, volume 21 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, Berlin, 1990.

[EMP97] H. Ehrig, A. Merten, and J. Padberg. How to Transfer Concepts of Abstract Data Types to Petri Nets. *EACTS Bulletin*, 62:106–104, 1997.

[Eng91] J. Engelfriet. Branching Processes of Petri Nets. *Acta Informatica*, 28:575–591, 1991.

[EO01a] H. Ehrig and F. Orejas. A Conceptual and Formal Framework for the Integration of Data Type and Process Modeling Techniques. In *Proc. GT-VMT 2001, ICALP 2001 Satellite Workshops*, pages 201–228, Heraclion, Greece, 2001.

[EO01b] H. Ehrig and F. Orejas. A Generic Component Concept for Integrated Data Type and Process Specification Techniques. Technical report, Technische Universität Berlin, FB Informatik, 2001.

[EP97a] H. Ehrig and J. Padberg. A Uniform Approach to Petri Nets. In Ch. Freksa, M. Jantzen, and R. Valk, Editors, *Foundations of Computer Science: Potential - Theory - Cognition*, pages 219–231. Springer, LNCS 1337, 1997.

[EP97b] H. Ehrig and J. Padberg. Introduction to Universal Parametrized Net Classes. In H. Weber, H. Ehrig, and W. Reisig, Editors, *MoveOn-Proc. der DFG-Forschergruppe "Petrinetz-Technologie"*, pages 39–51, Technische Universität Berlin, 1997. Forschungsberichte des Fachbereichs Informatik.

[EP97c] C. Ermel and J. Padberg. Formalization of Variables in Algebraic High-Level Nets. Technical Report 97-19, Technical University Berlin, 1997.

[EPR94] H. Ehrig, J. Padberg, and L. Ribeiro. Algebraic High-Level Nets: Petri Nets Revisited. In *Recent Trends in Data Type Specification*, pages 188–206. Springer Verlag, 1994. Lecture Notes in Computer Science 785.

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

[GR83] U. Goltz and W. Reisig. The Non-Sequential Behaviour of Petri Nets. In *Information and Computation*, volume 57, pages 125–147. Academic Press, 1983.

[Hof00] K. Hoffmann. Run Time Modification of Algebraic High Level Nets and Algebraic Higher Order Nets Using Folding and Unfolding Construction. In *Proceeding of the 3rd Internation Workshop Communication Based Systems*, pages 55–72. Kluwer Academic Publishers, 2000.

[Hof01] K. Hoffmann. Flexible Modellierung mit Algebraischen Higher Order Netzen. In *Proceeding of the Workshop Modellierung 2001*, pages 101–110.

[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.

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

[MMS97] J. Meseguer, U. Montanari, and V. Sassone. On the Semantics of Place/Transition Petri Nets. *Mathematical Structures in Computer Science*, 7:359–397, 1997.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science*, 13:85–108, 1981.

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

[PE01] J. Padberg and H. Ehrig. Introduction to Parametrized Net Classes. In H. Ehrig, G. Juhás, J. Padberg, and G. Rozenberg, Editors, *Advances in Petri Nets: Unifying Petri Nets*, LNCS. Springer, 2001.

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic High-Level Net Transformation Systems. *Mathematical Structures in Computer Science*, 5:217–256, 1995.

[PJE+01] J. Padberg, L. Jansen, H. Ehrig, E. Schnieder, and R. Heckel. Cooperability in Train Control Systems Specification of Scenarios Using Open Nets. *Journal of Integrated Design and Process Technology*, 5:3–21, 2001.

[Roz87] G. Rozenberg. Behaviour of Elementary Net Systems. In W. Brauer, W. Reisig, and G. Rozenberg, Editors, *Advances in Petri Nets 1986*, pages 60–94. Springer Verlag Berlin, LNCS 254, 1987.

[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.

[vGP95] R. v. Glabbeck and G. Plotkin. Configuration Structures. In *Proc. 10th LICS Symposium*. IEEE, 1995.

[Win87] G. Winskel. Petri Nets, Algebras, Morphisms, and Compositionality. *Information and Computation*, 72:197–238, 1987.

[Win88] G. Winskel. Event Structures. In W. Brauer, W. Reisig, and G. Rozenberg, Editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, pages 324 – 392. Springer, LNCS 255, 1988.