Towards a Modular Analysis of Coloured Petri Nets

Søren Christensen
Department of Computer Science, Aarhus University
Ny Munkegade, Bldg. 540
DK-8000 AARHUS C, Denmark
Phone: +45 86 12 71 88
Telefax: +45 86 13 57 25
E-mail: schristensen@daimi.aau.dk

Laure Petrucci
CEDRIC-IIE
18, Allée Jean Rostand
F-91025 EVRY Cedex, France
Phone: +33 (1) 60 77 97 40
Telefax: +33 (1) 60 77 96 99
E-mail: berthe@cnam.cnam.fr

June 1993

Abstract

The use of different High-level Petri net formalisms has made it possible to create Petri net models of large systems. Even though the use of such models allows the modeller to create compact representations of data and action, the size of models has been increasing. A large model can make it difficult to handle the complexity of the modelling as well as the analysis of the total model. It is well-known that the use of a modular approach to modelling has a lot of advantages. A modular approach allows the modeller to consider different parts of the system independently of one another and also to reuse the same module in different systems. A modular approach to analysis is also attractive. It often dramatically decreases the complexity of the analysis task.

In this paper, we present modular *CP*-nets. They are not intended to be used for practical modelling purposes, but they constitute a formal and general framework for discussing different ways of composing individual *CP*-nets called modules. Modular *CP*-nets allow us to study composition without restricting the structure of the individual modules. Modular *CP*-nets are quite simple and do not include syntactical sugar which is convenient and often necessary when modelling in practice. Instead, they have only a few but very general composition constructs.

The main result of the paper is the possibility of composing analysis results of the individual modules, in order to obtain results which are valid for the entire modular *CP*-net. For this purpose, we introduce place invariants at the level of modular *CP*-nets and we show how such place invariants can be obtained from those of the individual modules.

The reader of this paper is assumed to be familiar with the basic defmitions of *CP*-nets and the concept of place invariants. But it is not necessary to be familiar with hierarchical *CP*-nets.

1 Introduction

The use of High-Level Petri Nets, e.g., Predicate/Transition Nets and *CP*-nets, has been of great importance for the use of Petri Nets to model real systems. But it turns out that it is often necessary and convenient to separate Petri Nets into a number of related descriptions. The work on hierarchical *CP*-nets ([HJS90], [Jen91]) shows how a set of hierarchy concepts can be used to relate a set of *CP*-nets in a formal way. The purpose of hierarchical *CP*-nets has been to give the modeller a set of facilities which can ease the handling of large models. In contrast to this, we define modular *CP*-nets in order to be able to discuss composition concepts for *CP*-nets in a concise way. Reentrant nets, presented in [Che91], are coloured nets with a set of input places and a disjoint set of output places. Reentrant nets are meant to represent a protocol phase. Some equivalences allow to use different reentrant nets (with a same interface) as part of a model.

The reasons for defining modular *CP*-nets instead of using an existing model, e.g., hierarchical *CP*-nets, are twofold. First of all, we want a simplified model which does not include all the concepts which are useful when modelling with *CP*-nets. Instead, we only include the basic primitives needed in the discussion of *CP*-nets analysis. Secondly, we want to discuss composition concepts which are based both on the sharing of places and on the sharing of transitions. However, as explained below, no existing model has a formal definition and allows both relations between *CP*-nets.

Modular *CP*-nets consist of sets of formally related *CP*-nets, each *CP*-net is called a module. We consider two sorts of relations between modules which are quite natural and often used. The first construct can be described as a set of places sharing the same tokens. When a transition adds (respectively removes) a token to one of the places in me set, it is added to (respectively removed from) all the places in me set. We call this place fusion. For the second construct, we fuse sets of transitions. All transitions of a set occur as one indivisible action sharing the values assigned by a common binding. Place fusion and transition fusion were both introduced in [HJS90], but the formal definition of hierarchical *CP*-nets in [Jen91] only uses constructs based on place fusion. A formal model with constructs based on transition fusion can be found in [CD92]. The paper only considers place invariant analysis. An overview of different analysis methods of *CP*-nets can be found in [Jen92].

The rest of the paper is organized as follows:

In section 2, we informally introduce the notion of module *CP*-nets and the concepts of place sharing and transition sharing. An example shows the intuition behind place sharing and transition sharing.

In section 3 we explain how the analysis results for modules can be exploited to obtain results which apply to the entire module *CP*-net. The analysis method considered is place invariants calculus, which is applied to the examples of section 2.

In section 4, we introduce the formal definition of modular *CP*-nets and their behaviour, i.e., we define the enabling and occurrence rules. We also prove that each modular *CP*-net has a behaviorally equivalent *CP*-net.

Finally, section 5 contains the formal definitions of place invariants and place flows for modular *CP*-nets. We show that place invariants of module *CP*-nets correspond to place invariants of the equivalent *CP*-net. The main result is a constructive proof of how place flows of the individual modules can be combined to place flows of me module *CP*-net.

2 Modular *CP*-nets

In this section, we present two different ways of modelling a problem, one using place sharing, and the other using transition sharing. The example described is a variation of the resource allocation system from [Jen91]. In the next section, the resource allocation examples are used to show how analysis results of modules can be composed. This means that properties of a modular *CP*-net can be proved by means of formal analysis of the individual modules.

The resource allocation example has a set of processes which share a common pool of resources. There are two different kinds of processes, called p-processes and q-processes, and three different kinds of resources: r-resources, s-resources and t-resources. Each process is cyclic and during the individual parts of its cycle, the process needs to have exclusive access to a varying amount of the resources. For each process, we have an integer value counting the number of process cycles. We use the following definition of colours: $P = \{p, q\}$, $I = \text{Integer}$, $U = P \times I$ and $E = \{e\}$. We use a variable x of type P and a variable i of type I. A multi-set containing two p tokens with count 5 and one q token with count 3 will be denoted by $2'(p, 5) + 1'(q, 3)$. The p-processes can be in four different states, while q-processes can be in five different states. In the initial state, there are 2 p-processes and 3 q-processes, plus 1 r-resource, 3 s-resources and 2 t-resources. The *CP*-net is presented in Fig. 1. It is so small that we would not decompose it in practice, but it can still be used to introduce the basic concepts of modular *CP*-nets.

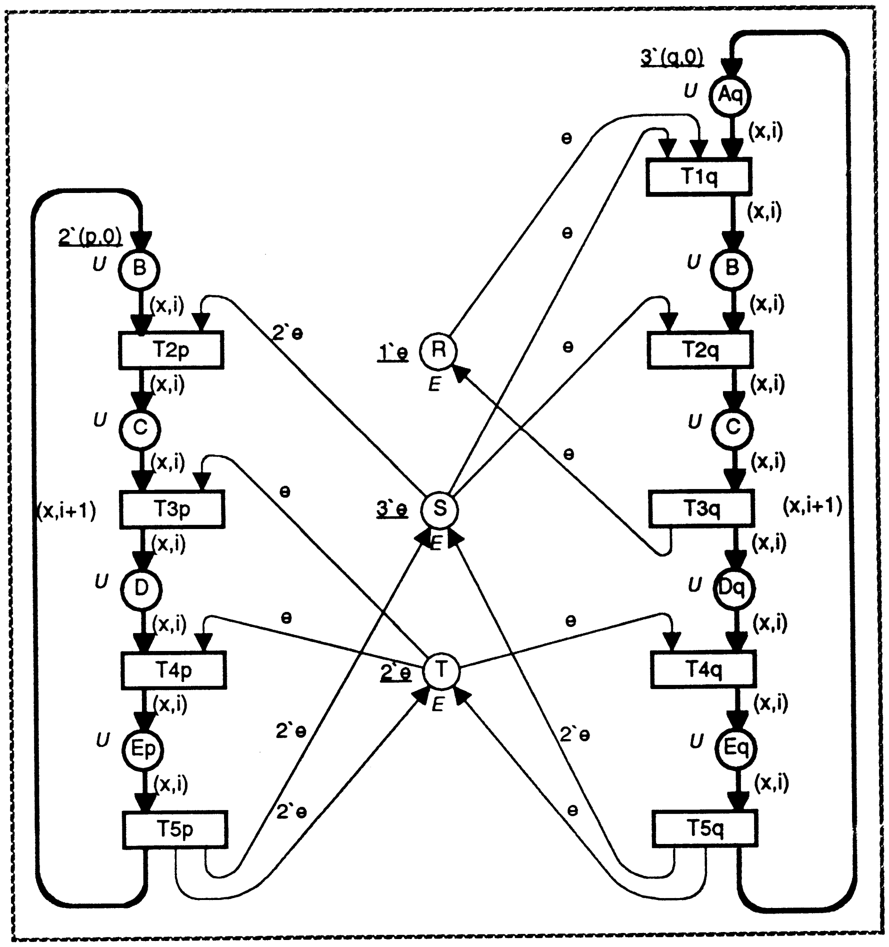

Figure 1: Example of *CP*-net.

A first possibility consists in modelling separately the p-processes and the q-processes. This leads to two modules, as shown in Fig. 2, each of them describing the interaction between processes of one kind and the resources. The modules are composed by fusion of the two shared resource places, S and T. In Fig. 2, the places to be fused together have the same name. In a practical modelling tool we would need more elaborated techniques to identify members of a given fusion set. All places in a place fusion set must have the same colour, and the same initial marking.

You can view all places of a place fusion set as being representatives of the same underlying place. This means that they share the same marking:

when a token is added to a place which belongs to a place fusion set, all places of the place fusion set will have the same token added. When a token is removed from a place which belongs to a place fusion set, all places of the place fusion set will have the same token removed. The fusion of the resolve places, S and T, ensures that the modular *CP*-net of Fig. 2 has exactly the sue behaviour as the *CP*-net of Fig. 1.

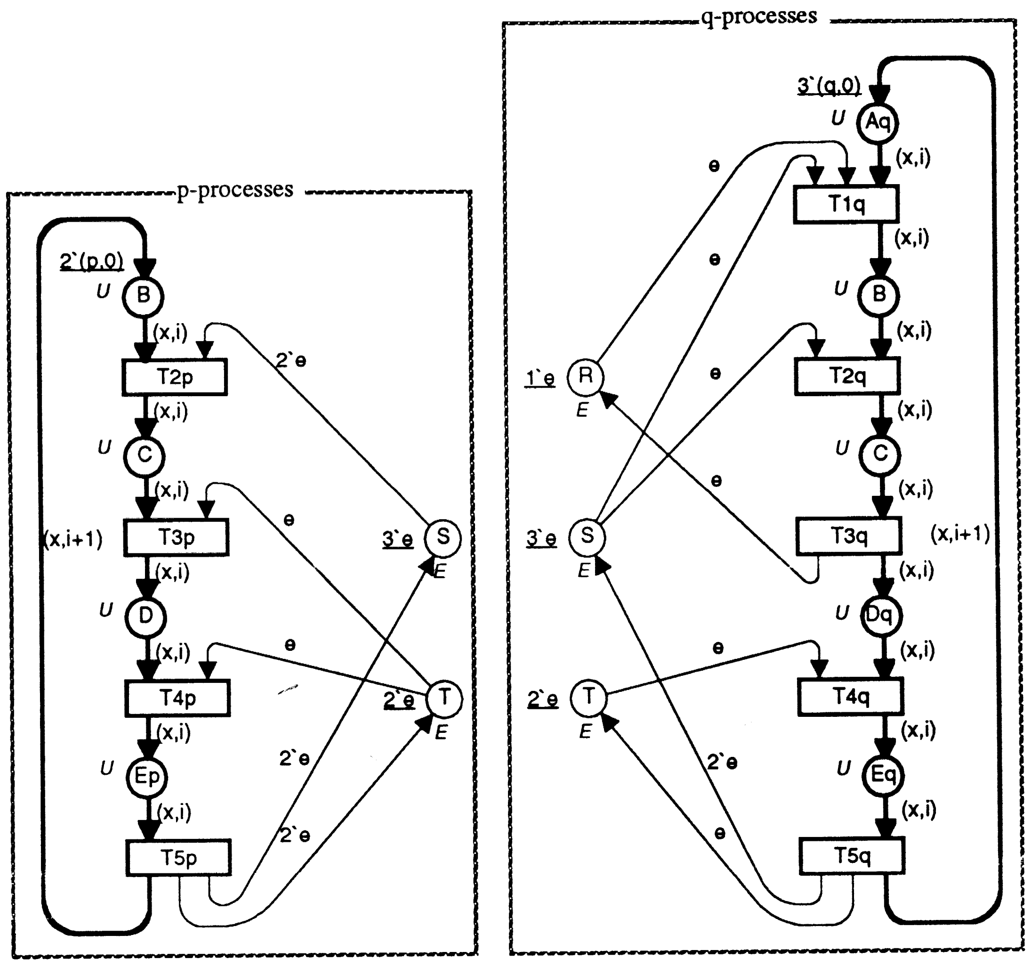

Figure 2: Modular *CP*-net with 2 modules and 2 place fusion sets with 2 members each.

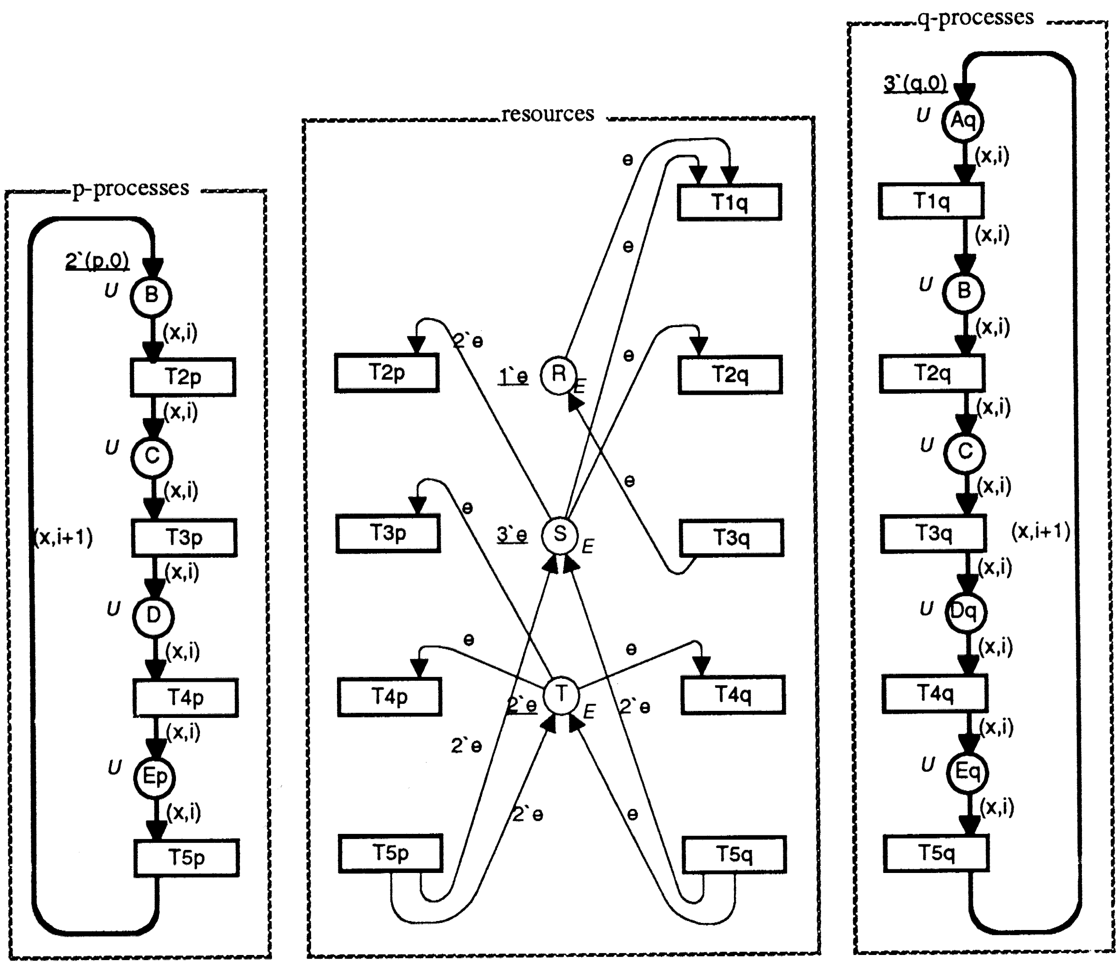

Figure 3: Modular *CP*-net with 3 modules and 9 transition fusion sets with 2 members each.

Another way of modelling the resource allocation system is to separate the cycle of *p*-processes, the one of *q*-processes and the use of resources, as shown in Fig 3. The three modules share transitions, corresponding to synchronous actions. Transitions having the same names belong to the same fusion set. This means that we have 9 transition fusion sets with 2 members each. Each transition of a transition fusion set describes a part of a more complex action and all parts must occur together. We say that a transition fusion set is enabled if we can specify a set of bindings such that all transitions in the fusion set are enabled for this set of bindings. A variable may be shared by several of the transitions and will be bound to the same value for all of these. The change produced by the occurrence of a transition fusion set for a given binding is the sum of changes produced by all the transitions of the fusion set.

A transition can describe an action which is a basic part of a number of independent actions. This means that a transition can be a member of several transition fusion sets. Since the behaviour of the three transitions $T3p$, $T4p$ and $T4q$ is identical, we could include only one of the three transitions and have it as a member of three transition fusion sets. If our main concern was guidelines for modelling we would have done this, but Fig. 3 corresponds to a straightforward separation of the original *CP*-net.

The modular *CP*-net of Fig. 3 has exactly the same behaviour as the *CP*-net in Fig 1. Each transition fusion set of the modular *CP*-net corresponds to exactly one transition of the *CP*-net.

We have presented two examples of modular *CP*-nets having me same behaviour The first one was an example of modules related by place fusion while the second one used transition fusion. In general, both place fusion and position fusion can coexist within a module *CP*-net.

In the next section, we will present a module approach of place invests calculus.

3 Place Invariants of Modular *CP*-nets

All analysis methods extract information about properties of a *CP*-net in a condensed way. Place invert use me idea of mapping the markings of all places into a common colour set. The mapping is done by means of weights attached to the places. A weight is a function which specifies the information we want to extract from the markings of a place. Since we want to compare the information extracted from the markings of different places, the weights must map from markings of a plate into a common range. A weight-function maps each place into such a weight. We say that a weight-function determines a place invariant if the multi-set sum of the weighted markings of places is constant for all reachable markings. It is often the case that a single place invariant ignores the marking of some places. This is done by assigning a zero function to the places which should be ignored.

3.1 Place Invariants of the *CP*-net

For the example shown in Fig. 1, we find several place invariants. One of the place invariants extracts the identity of the processes from the places called: Aq, Bq, Cq, Dq and Eq, and no information from the others. This means that a projection function is used as the weight for Aq, Bq, Cq, Dq and Eq, and a zero function is the weight used for all other places. We can show that the sum of the weighted markings is constant for all reachable markings. This means that the set of q-processes does not change, only their state changes. Instead of checking all reachable markings, we can also check that the weighted sum of tokens consumed by each binding for each transition is equal to the weighted sum of tokens produced, We say that a weight-function having this property defines a place flow. It can be proved that the place flow property is sufficient to ensure that the weight unction determines a place invest.

In our notation of weight functions we use the following notation: places having a zero weight are simply left out, the identity function is implicit, the Pr (Projection) function maps multi-sets of pairs of $P \times I$ into multi-sets of P, the Ig (Ignore colour) function maps a multi-set of size s into $s'e$ and we use the noes of the places to refer to the marking of the place, e.g., we write Bp instead of $M(Bp)$. For the example in Fig. 1, we have five place invariants:

W_1 $Pr(Bp + Cp + Dp + Ep) = 2'p.$
W_2 $Pr(Aq + Bq + Cq + Dq + Eq) = 3'q.$
W_3 $R + Ig(Bq + Cq) = 1'e.$
W_4 $S + Ig(Bq) + 2 * Ig(Cp + Dp + Ep + Cq + Dq + Eq) = 3'e.$
W_5 $T + Ig(Dp + Eq) + 2 * Ig(Ep) = 2'e.$

We can construct other place invests but all of the above place invariants can easily be interpreted in terfns of the *CP*-net: as an example, W_1 shows that all the p-processes are in one of the states represented by Bp, Cp, Dp or Ep. W_3 shows that the R resources are either free (i.e. in state R) or occupied by a q-process in state Bq or Cq. Using the information from the five place invests above, it is straightforward to prove that the system is deadlock free and similar behavioural properties.

3.2 Plate Invariants of the Modular *CP*-net with Place Sharing

In Fig. 2. we have two modules, one for the p-processes and one for the q-processes. The two modules are related trough two place fusion sets, i.e., through the s-resources and t-resources. For the p-processes, we have three place invariants:

W_{p1}	$Pr(Bp + Cp + Dp + Ep) = 2'p.$
W_{p2}	$S + 2 * Ig(Cp + Dp + Ep) = 3'e.$
W_{p3}	$T + Ig(Dp) + 2 * Ig(Ep) = 2'e.$

For the q-processes, we have for place invariants:

W_{q1}	$Pr(Aq + Bq + Cq + Dq + Eq) = 3'q.$
W_{q2}	$R + Ig(Bq + Cq) = 1'e.$
W_{q3}	$S + Ig(Bq) + 2 * Ig(Cq + Dq + Eq) = 3'e.$
W_{q4}	$T + Ig(Eq) = 2'e.$

Is it possible to construct place invariants of the total system from the place invariants of the individual modules?

The place invariants W_{p1}, W_{q1} and W_{q2} do not include any places which are shared, so W_{p1}, W_{q1} and W_{q2} are place invariants of the total system, independently of the context of the other modules in the modular *CP*-net. The rest of the place invariants have non-zero weights for some of the shared places. In this situation we can only combine the place invariants if the weight functions assign the same weight to the shoed places. This means that we can combine W_{p2} and W_{q3}, because they both have an identity weight for S and a zero weight for T. Analogously, we can combine W_{p3} and W_{q4} because they both have an identity weight for T and a zero weight for S. From the place invariants of the individual modules we deduce the following place invariants of the modular *CP*-net:

W_{p1}	$Pr(Bp + Cp + Dp + Ep) = 2'p.$
W_{q1}	$Pr(Aq + Bq + Cq + Dq + Eq) = 3'q.$
W_{q2}	$R + Ig(Bq + Cq) = 1'e.$
$W_{p2} + W_{q3}$	$S + Ig(Bq) + 2 * Ig(Cp + Dp + Ep + Cq + Dq + Eq) = 3'e.$
$W_{p3} + W_{q4}$	$T + Ig(Dp + Eq) + 2 * Ig(Ep) = 2'e.$

These five place invariants correspond to linear independent place invariants of the equivalent *CP*-net.

From the above, we see that a set of place invariants of the modules can be combined to cover the total system if they have the same weights for places which are shared. This is not true in the general case, but if we restrict ourselves to combining place flows it is legal. This will be detailed in the formal definition of invariants (section 5). If the weight functions do not match, in the sense described above we may sometimes obtain a matching by scaling one of the weight functions. The scaling is done by "multiplying" all of the weights by a common function.

3.3 Plain Invariants of the Modular *CP*-net with Transition Sharing

For the example shown in Fig. 3, we have three modules, one for the p-processes, one for the q-processes and one for the resources. The three modules are related through transition fusion for each pair composed of a transition in p-processes or q-processes and the corresponding transition in the resource module. If we view the modules independently of their context we can find a set of place invariants of the individual *CP*-nets. For the p-processes, we have one place invariant:

$$W_p: \quad Pr(Bp + Cp + Dp + Ep) = 2'p.$$

For the q-processes, we have one place invariant:

$$W_q: \quad Pr(Aq + Bq + Cq + Dq + Eq) = 3'p.$$

And the resource sharing module has only the trivial zero place invariant. Is it possible to construct place invariants of the total system from the place invariants of the modules?

In this case we have no shared places and this means that W_p and W_q both are invariants of the entire system. However, it should be obvious that we cannot construct all the place invariants from those of the individual modules. The problem is that we demand too much of the weight functions. We demand that each transition leaves the invariant unchanged, while it would be sufficient to demand that each transition fusion set does this. In order to relax the conditions of the weight functions for the modules, we introduce the notion of flow preservation. We say that a transition is flow preserving if and only if for any binding it preserves the invariant. Then we are able to check that the non-fused transitions are flow preserving for each module. And that transition fusion sets are flow preserving across modules, i.e., that the transitions *together* preserve the invariant.

The above examples allowed us to show the intuition behind module *CP*-nets and their alive the concepts analysis by means of place invariants. In the next sections, we formalize the concepts presented up to now. In section 4, we give the formal definition of a modular *CP*-net and of the equivalent *CP*-net. In section 5, we extend the notion of place invariants and place flows to modular *CP*-nets.

4 Formal Definition of Modular *CP*-nets

Before defining modular *CP*-nets, we recall the definition of Coloured Petri Nets and the basic concepts used in this definition. We use the notations of [Jen92], but equivalent definitions can be found in [Jen91]. To give the abstract definition of *CP*-nets, it is not necessary to fix the concrete syntax in which the modeller writes the net expressions, and we shall only assume that such a syntax exists, making it possible in an unambiguous way to talk about:

- The *elements of a type*, T. The set of all elements in T is denoted by the type name T itself. The set of multi-sets over T is denoted by T_{MS}.
- The *type of a variable*, v—denoted by $Type(v)$.
- The *type of an expression*, $expr$—denoted by $Type(expr)$.
- The *set of variables in an expression*, $expr$—denoted by $Var(expr)$.
- A *binding of a set of variables*, V—associating with each variable $v \in V$ an element $b(v) \in Type(v)$.
- The *value obtained by evaluating an expression*, $expr$, *in a binding*, b—denoted by $expr < b >$. $Var(expr)$ is required to be a subset of the variables of b, and the evaluation is performed by substituting for

each variable $v \in \mathit{Var}(\mathit{expr})$ the value $b(v) \in \mathit{Type}(v)$ determined by the binding. An expression *without* variables is said to be a *closed* expression. It can be evaluated in all bindings, and all evaluations give the same value—which we often shall denote by the expression itself. This means that we simply write "*expr*" instead of the more pedantic "*expr* <>".

Now we are ready to define *CP*-nets. We use $\mathbb{B}$ to denote the boolean type (containing the elements $\{\mathsf{false}, \mathsf{true}\}$, and having the standard operations from propositional logic), and when *Vars* is a set of variables, we use $\mathit{Type}(\mathit{Vars})$ to denote the set of types $\{\mathit{Type}(v) \mid v \in \mathit{Vars}\}$.

Definition4.1 ([Jen92], Def. 2.5)
A *CP*-**net** is a tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ satisfying the requirements below:

- (i) Σ is a finite set of types, called **colour sets**.
- (ii) P is a finite set of **places**.
- (iii) T is a finite set of **transitions.**
- (iv) A is a finite set of **arcs** such that:
  - $P \cap T = P \cap A = T \cap A = \emptyset$.
- (v) N is a **node** function. It is defined from A into $P \times T \cup T \times P$.
- (vi) C is a **colour** function. It is defined from P into Σ.
- (vii) G is a **guard** function, It is defined from T into expressions such that:
  - $\forall t \in T : [\mathrm{Type}(G(t)) = \mathbb{B} \ \wedge \mathit{Type}(\mathit{Var}(G(t))) \subseteq \Sigma]$
- (viii) E is an **arc expression** function. It is defined from A into expressions such that:
  - $\forall a \in A : [\mathit{Type}(E(a)) = C(p(a))_{MS} \wedge \mathit{Type}(Var(E(a))) \subseteq \Sigma]$

  where $p(a)$ is the place of $N(a)$.
- (ix) I is an **initialization** function. It is defined from P into closed expressions such that:
  - $\forall p \in P : [\mathit{Type}(I(p)) = C(p)_{MS}]$.

We denote the set of arcs connected to a node x by $A(x)$. The set of variables associated with a position t is denoted by $\mathit{Var}(t)$ and defined by:

$\forall t \in T : \mathit{Var}(t) = \{v \mid v \in \mathit{Var}(G(t)) \vee \exists a \in A(t) : v \in$

$Var(E(a))\}$.

Definition 4.2([Jen92], Def. 2.6)
A **binding** of a transition t is a function b defined on $Var(t)$ such that:
(i) $\forall v \in Var(t) : b(v) \in Type(v)$.
(ii) $G(t) <b>$.
By $B(t)$ we denote the set of all bindings for t.

To talk about the elements of the entire Module *CP*-net with modules in a set S, we use the notations

$$P = \bigcup_{s \in S} P_s \quad T = \bigcup_{s \in S} T_s \quad A = \bigcup_{s \in S} A_s$$

The disjointness of the net elements means that without ambiguity we can use a "global" colour set function $C \in [P \rightarrow \Sigma]$ instead of the "local" colour set functions $C_s \in [P_s \rightarrow \Sigma_s]$. me global function is defined from the local functions, in the following way:

$$\forall s \in S \; \forall p \in P_s : C(p) = C_s(p).$$

Analogously, we can define global versions of the *node* function N, the *guard* function G, the *arc expression* function E, and the *initialization* function I. It also means that we can use $p(a), t(a), s(a), d(a), A(x_1, x_2), A(x), In(x)$, $Out(x)$, $X(x)$ and $E(x_1, x_2)$ – in exactly the same way as for *CP*-nets. When x_1 and x_2 belong to different modules $A(x_1, x_2) = \emptyset$ and $E(x_1, x_2) = \emptyset$.

Some motivation and explanation of the individual parts of the definition are given immediately below it, and it is recommended to read both in parallel.

Definition 4.3
A **Modular** ***CP*-net** is a triple $MCPN = (S, PF, TF)$, satisfying the following requirements:
(i) S is a finite set of **modules** such that:
- Each module, $s \in S$, is a *CP*-net: $M_s = (\Sigma_s, P_s, T_s, A_s, N_s, C_s, G_s, E_s, I_s)$.

• The sets of net elements are pairwise disjoint:
$\forall s_1, s_2 \in S : [s_1 \neq s_2 \Rightarrow (P_{s_1} \cup T_{s_1} \cup A_{s_1}) \cap (P_{s_2} \cup T_{s_2} \cup A_{s_1}) = \emptyset]$.

(ii) $PF \subseteq 2^P$ is a finite set of **place fusion** sets such that:
• Members of a place fusion set have identical colour sets and equivalent initialization expressions.
$$\forall p_1, p_2 \in pf : [C(p_1) = C(p_2) \wedge I(p_1) = I(p_2)]$$

(iii) $TF \subseteq 2^T$ is a finite set of **transition fusion sets.**

(i) A module *CP*-net contains a finite set of modules, each of them being a *CP*-net. These modules must have disjoint sets of places, positions and arcs.

(ii) Each place fusion set is a set of plates to be used together. 2^P denotes the set of all sub-sets of places. We demand that all elements of a place fusion set have the same colour set and that they have equivalent initial markings. By external places $EP \subseteq P$ we denote the set of all places which are members of a place fusion set and by internal places, $IP = P - EP$, we denote all non-fused places. It should be noted that we, in contrast to [HJS90], do not demand the place fusion sets to be disjoint.

(iii) Each transition fusion set is a set of transitions to be fused together. By external transition $ET \subseteq T$ we denote the set of all transitions which are members of a transition fusion set and by internal transitions, $IT = T - ET$, we denote all non-fused transitions. It should be noted that wee in contrast to [HJS90], do not demand the transition fusion sets to be disjoint.

In Def. 4.4, we introduce place groups and transition groups. The notion of place groups corresponds to the notion of place instance groups for hierarchical *CP*-nets ([Jen92, Def. 3.5]).

Definition 4.4
A **place group** $pg \subseteq P$ is an equivalence class of the smallest equivalence relation containing all pairs $(p_1, p_2) \in P \times P$ where
$\exists pf \in PF : p_1, p_2 \in pf$.
A **transition group** $tg \subseteq T$ consists of either a single non-fused transition t or all the members of a transition fusion set $tf \in TF$.

Place groups and transition groups are defined very differently since a place can be a member of at most one plate group while a transition can be a member of several transition groups. Place groups form a petition of the set of places. This means that each place p is a member of one and only one place grump which shall be denoted $[p]$. Note that all place groups and transition groups have at least one element. In the followings we use names with a prime to denote place groups and transition groups, e.g. p' and t'. From Def. 4.3 (ii) and 4.4 we know that all places of a place group will have the same colour set and equivalent initial markings, this allows us to talk about $C(p')$ and $I(p')$ without being ambiguous. We define: $\forall p' = [p] \in PG : C(p') = C(p)$ and $I(p') = I(p)$.

Next, we define the set of variables associated with a transition group, the binding of a transition group and the guard of a transition group.

Definition 4.5
A **binding** of a transition group t' is a function b defined on the variables of the transition group, $Var(t') = \bigcup_{t \in t'} Var(t)$, such that:
(i) $\forall v \in Var(t') : b(v) \in Type(v)$.
(ii) $\forall t \in t' : G(t) <b>$.
We denote the conjunction of guards of a transition group t' by $G(t')$, and the set of all bindings by $B(t')$.

A binding will assign only one value for a variable, i.e. a variable name will refer to the same value for all transitions in a transition group.

Next, we extend the arc function A to handle place groups and transition groups:

$$A(x', y') = \{a \in A \mid \exists x \in x' \exists y \in y' : N(x, y) = a\}.$$

Then the expression function E is extended from arcs to place groups and transition groups. The summation is well-defined because all the involved expressions have the same type, according to Def. 4.1 (viii) and Def. 4.3 (ii):

$$\forall (x_1, x_2) \in (PG \times TG \cup TG \times PG) : E(x', y') = \sum_{a \in A(x', y')} E(a).$$

Now, we define token elements, bindings elements, markings and steps for modular CP-nets. This is done in a similar way as for hierarchical CP-nets.

Definition 4.6
A **token element** is a pair (p', c) where $p' \in PG$ and $c \in C(p')$, while a **binding element** is a pair (t', b) where $t' \in TG$ and $b \in B(t')$. The set of all token elements is denoted by TE while the set of all binding elements is denoted by BE.

A **marking** is a multi-set over TE while a **step** is a *non-empty* and *finite* multi-set over BE. The initial marking M_0 is the marking which is obtained by evaluating the initialization expressions:
$\forall (p', c) \in TE : M_0(p', c) = I(p')(c)$.
The set of all markings and steps are denoted by $\mathbb{M}$ and $\mathbb{Y}$, respectively.

The enabling rule of a module *CP*-net can now be expressed. The inequality used to compare a value to a marking is the inequality of multi-sets.

Definition 4.7
A step Y is **enabled** in a marking M iff the following property is satisfied:
$\forall p' \in PG : \sum_{(t',b)\in Y} E(p', t') <b> \leq M(p')$.
When a step Y is enabled in a marking M_1 it may **occur**, changing the marking M_1 to another marking M_2, defined by:
$\forall p' \in PG : M_2(p') = (M_1(p') - \sum_{(t',b)\in Y} E(p', t') <b>) +$
$\sum_{(t',b)\in Y} E(t', p') <b>$.

We say that M_2 is **directly reachable** from M_1 by the occurrence of step Y, which we also denote by: $M_1[Y \succ M_2$.

Next, we show that each modular *CP*-net has a behavioural equivalent *CP*-net.

We denote the source of an arc a by $s(a)$, the place connected to a by $p(a)$ and the transition connected to a by $t(a)$. Some motivation and explanation of individual parts of the definition of the equivalent *CP*-net is given immediately below it, and it is recommended to read both in parallel.

Definition 4.8
Let a modular *CP*-net $MCPN = (S, PF, TF)$ be given. Then we define the **equivalent CP-net** to be $CPN = (\Sigma^*, P^*, T^*, A^*, N^*, C^*, G^*, E^*, I^*)$ where:

(i)	$\Sigma^* = \Sigma$.
(ii)	$P^* = PG$.
(iii)	$T^* = TG$.
(iv)	$A^* = \{(a, t') \in A \times TG \mid t(a) \in t'\}$.
(v)	$\forall a^* = (a, t') \in A^*$: $[\ s(a) \in P \Rightarrow N^*(a^*) = ([p(a)], t') \wedge$ $s(a) \in T \Rightarrow N^*(a^*) = (t', [p(a)])]$.
(vi)	$\forall p^* \in P^* : C^*(p^*) = C(p^*)$.
(vii)	$\forall t^* \in T^* : G^*(t^*) = G(t^*)$.
(viii)	$\forall a^* = (a, t') \in A^* : E^*(a^*) = E(a)$.
(ix)	$\forall p^* \in P^* : I^*(p^*) = I(p^*)$.

Note that all components of the constructed *CP*-net are valid.

(i) The set of colours sets of the equivalent *CP*-net is equal to the union of the colour sets of the modules.

(ii) The equivalent *CP*-net has one place for each place group.

(iii) The equivalent *CP*-net has one transition for each transition group.

(iv) The transition of an arc may belong to several transition groups. When this is the case, we get a copy of the arc for each transition group.

(v) The nodes of an arc can be determined from the transition group attached to it and from the place of the original arc. From (iv), we know that $t(a) \in t'$.

(vi) From Def. 4.3(ii) and 4.4 we know that all places of a place group have the same colour set and we know that all place groups have at least one member. The colour set of a place group is determined by one of the members of the place group.

(vii) The guard of a transition group is the conjunction of the guards of the transitions which are members of the transition group.

(viii) The expression associated with an arc is the same as the expression of the original arc.

(ix) From Def. 4.3(ii) and 4.4 we know that all places of a place group have the same initial marking and we know that all place groups have at least one member. The initial marking of a place group is determined by one of the members of the place group.

In section 2, we claimed that the presented modular *CP*-nets and the *CP*-net, given as examples, were equivalent according to the behaviour. This can be checked using Def. 4.7.

The following theorem shows that a modular *CP*-net and its equivalent *CP*-net have the same behaviour. All names that refer to the equivalent *CP*-net are marked by an asterisk, e.g., M_0 refers to the initial marking of the modular *CP*-net and M_0^* to the initial marking of its equivalent *CP*-net.

Definition 4.9
Let *MCPN* be a modular *CP*-net and let CPN^* be the equivalent *CP*-net. Then we have the following properties:
(i) $\mathbb{M} = \mathbb{M}^* \wedge M_0 = M_0^*$.
(ii) $\mathbb{Y} = \mathbb{Y}^*$.
(iii) $\forall\ M_1, M_2 \in \mathbb{M}, \forall\ Y \in \mathbb{Y} : M_1[Y \succ_{MCPN} M_2 \Leftrightarrow M_1[Y \succ_{CPN^*} M_2$.

Proof:

Property (i): From [Jen92], Def. 2.7, we have $\mathbb{M}^* = TE^*_{MS}$ where TE^* consists of all pairs (p^*, c) with $p^* \in P^*$ and $c \in C^*(p^*)$. From Def. 4.69, we have $\mathbb{M} = TE_{MS}$ where TE consists of all pairs (p', c) with $p' = [p] \in PG$ and $c \in C(p)$. Thus, it is sufficient to prove that $P^* = PG$ and $C^*(p^*) = C(p)$, but this follows from Def. 4.8 (ii) and (vi).
Next, let us prove that the two initial markings are identical. From Def. 4.6, we have:

$(*)\ \ \forall (p', c) = ([p], c) \in TE : M_0(p', c) = (I(p))(c).$

From [Jen92], Def. 2.7, we have:

$\forall (p^*, c) \in TE^* : M_0^*(p^*, c) = (I^*(p^*))(c),$

which by Def. 4.8 (ii) and Def. 4.6 is equivalent to:

$\forall (p', c) = ([p], c) \in TE : M_0^*(p', c) = (I^*(p'))(c),$

which by Def. 43 (ix) is equivalent to:

$\forall (p', c) = ([p], c) \in TE : M_0^*(p', c) = (I(p))(c),$

which has the same form as (*).

Property (ii): From [Jen92], Def. 2.7. we have that $\mathbb{Y}^*$ consists of all non-empty and finite multi-sets in BE^*_{MS} where BE^* consists of all pairs (t^*, b) with $t^* \in T^*$ and $b \in B^*(t^*)$. From Def. 4.6, we have that $\mathbb{Y}$ consists of all non-empty and finite multi-sets in BE_{MS} where TE consists of all pairs (t', b) with $t' \in TG$ and $b \in B(t')$. Thus it is sufficient to prove that $T^* = TG$ and $B^*(t^*) = B(t)$, but this follows from Def. 4.8 (iii), (iv), (v), (vii) and (viii).

Property (iii): First, we prove that the enabling rules coincide, i.e.:

$M_1[Y \succ_{MCPN} \Leftrightarrow M_1[Y \succ_{CPN}$.

From Def. 4.7 it follows that $M_1[Y \succ_{MCPN}$ iff:

$(**) \quad \forall p' \in PG : \sum_{(t',b)\in Y} E(p', t') <b> \leq M_1(p')$.

From [Jen92], Def. 2.7, it follows that $M_1[Y \succ_{CPN}$ iff:

$\forall p^* \in P^* \sum_{(t',b)\in Y} E^*(p^*, t) <b> \leq M_1(p^*)$.

which by Def. 4.3 (ii)+(iii) is equivalent to:

$\forall p' \in PG \sum_{(t',b)\in Y} E^*(p', t') <b> \leq M_1(p')$.

which by Def. 4.8 (iv) + (vi) + (ix) and the extension of E^* from A^* to $(PG \times TG \cup TG \times PG)$, is equivalent to:

$\forall p' \in PG \sum_{(t',b)\in Y} E(p', t') <b> \leq M_1(p')$.

which is identical to $(**)$.

Next we must prove that the occurrence rules coincide, i.e.:

$M_1[Y \succ_{MCPN} M_2 \Leftrightarrow M_1[Y \succ_{CPN} M_2$.

This part of the proof can be structured just like the part regarding enabling rules and we will not include it. ♦

We have defined modular *CP*-nets which allow a user to express relationship between places and transitions in individual *CP*-nets called modules. The behaviour of module *CP*-nets was detailed. In the next section, we will discuss how the place invariants analysis method for *CP*-nets can be extended to modular *CP*-nets.

5 Place Invariant Analysis

In this section we show how the concepts of place invariants and place flows can be extended to modular *CP*-nets. Place invariants can be used in the proofs of properties of a *CP*-net, e.g. to show that there are no dead markings. In this paper, we focus on the concepts of place invariants and place flows, more than on the use of invariants in the proof of properties of *CP*-nets. It is possible to find examples of the use of place invariants in e.g. [Jen81], [Jen86], [Jen91] and [Jen92].

5.1 Place Invariarnts of *CP*-nets

In this subsection, we define the concepts of place invariants and place flows. But we first define some basic concepts.

The example in Fig. 4 shows a transition $T1$ which, for any binding, produces two identical tokens, one for place A and one for place B. If we want to specify a weight for A and B such that the weighted sum of tokens is constant, we need to allow both positive and negative weights. The easiest and most general way to obtain this is to replace multi-sets by weighted sets.

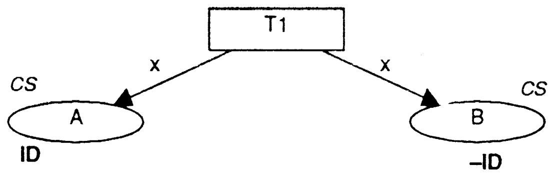

Figure 4: It can be convenient to have both positive and negative weights.

A **weighted set** over a non-empty set S, is defined in exactly the same way as a multi-set—except that we now also allow negative coefficients. This means that we can always subtract two weighted sets over the same set S from each other, and it also means that scalar-multiplication with negative integers can be allowed. The set of all weighted sets over S is denoted by S_{WS}. Weighted sets have properties which are analogous to those of multi-sets. In particular, we say that a function $W \in [A_{WS} \rightarrow B_{WS}]$ is **linear** iff:

$W(w_1 + w_2) = W(w_1) + W(w_2)$

for all weighted-sets $w_1, w_2 \in A_{WS}$. The set of linear functions in $[A_{WS} \rightarrow B_{WS}]$ is denoted by $[A_{WS} \rightarrow B_{WS}]_L$

Definition 5.1.1
Let $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ be a CP-net.

(i) W is a **weight function** with **range** $R \in \Sigma$ iff:
$\forall p \in P : W(p) \in [C(p)_{WS} \rightarrow R_{WS}]_L$.

(ii) The weight function W determines a **weighted marking**:
$\forall M \in \mathbb{M} : W(M) = \sum_{p \in P} W(p)(M(p))$.

(iii) The weight function W determines a **place invariant** iff:
$\forall M \in [M_0 >: W(M) = W(M_0)$.

A weight function maps each place p into a weight $W(p)$ which is a linear function—mapping from weigthed-sets over the colour set of p into weigthed-sets over some colour set R. R is common to all weights of W and we call this the range of W. For a given marking we calculate the weighted marking as the sum of the weights of the individual places. The weight function W determines an invariant iff all reachable markings have the same weighted sum.

In the following, we use names with a double prime to denote subsets, e.g. $T'' \subseteq T$.

Definition 5.1.2
Let $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ be a CP-net.

(i) A transition t is **flow preserving** for a weight function W iff:
$\forall b \in B(t) : \sum_{p \in P} W(p)(E(p,t) <b>) =$
$\sum_{p \in P} W(p)(E(t,p) <b>)$.

(ii) The weight function W is a **place flow of T$''$** iff all transitions of T'' are flow preserving.

(iii) The weight function W is a **place flow** iff it is a place flow of T.

We say that a transition is flow preserving for a weight function W iff each binding removes—when W is taken into account—the same set of tokens as it adds. In practice we do not need to sum through all places, it is sufficient to sum through $In(t)$ for the first sum and through $Out(t)$ for the second sum.

All weights are liner functions. This means that any linear combination of two place flows is a place flow, e.g. if W_1 and W_2 are place flows, with identical range, and $z_1, z_2 \in \mathbb{Z}$ then $z_1 * W_1 + z_2 * W_2$ is a place flow. A zero weight is a function which maps any weighted-set to the empty set. The weight function which assigns zero weights to all places is always a place flow. We say that a place p is included in W if $W(p)$ is a non-zero function. Similar remarks apply to place invariants.

The main reason for introducing place flows is the difficulty to check place invariants on the total set of reachable states. Place flows can be checked on the structure of the *CP*-net.

The following theorem describes the relationship between place invariants and place flows. A binding element is said to be dead when it can never occur.

Definition 5.1.3
Let a *CP*-net be given and let W be a weight function.
(i) W is a place flow $\Rightarrow W$ determines a place invariant.
(ii) If no binding elements are dead:
W is a place flow $\Leftrightarrow W$ determines a place invariant.

Proof: The theorem is part of the classical theory for invariant analysis. For *CP*-nets a proof of (i) can be found in [Jen81] and [Jen86]. The proof of (ii) is straightforward. ♦

5.2 Place Invariants of Modular *CP*-nets

In this section, we show how the formal definitions of place invariants and place flows can be given for modular *CP*-nets, This sub-section is structured like the previous one and it should be easy to compare the definitions given for *CP*-nets and Modular *CP*-nets.

Definition 5.2.1
Let a $MCPN = (S, PF, TF)$ be a modular *CP*-net.
(i) W is a **weight function** with **range** $R \in \Sigma$ iff:
$\forall p' \in PG : W(p') \in [C(p')_{WS} \rightarrow R_{WS}]_L$.
(ii) The weight function W determines a **weighted marking**:
$\forall M \in \mathbb{M} : W(M) = \sum_{p' \in PG} W(p')(M(p'))$.

(iii) The weight function W determines a **place invariant** iff:
$\forall M \in [M_0>: W(M) = W(M_0)$.

Weight functions of Module CP-nets map place groups to weights, just like weight functions of CP-nets map places to weights. If the Module CP-net contains no place fusion sets, the definitions are totally equivalent. Weighted markings and place invariants are also generalized to work on place groups.

Definition 5.2.2
Let a $MCPN = (S, PF, TF)$ be a modular CP-net.

(i) A position group t' is **flow preserving** for a weight function W iff:
$\forall b \in B(t') : \sum_{p' \in PG} W(p')(E(p',t') <b>= \sum_{p' \in PG} W(p')(E(t',p') <b>)$.

(ii) The weight function W is a **place flow of** TG'' iff all transition groups of TG'' are flow preserving.

(iii) The weight function W is a **place flow** iff it is a place flow of TG, i.e., all transition groups are flow preserving.

The concept of flow preserving is defined for transition groups, this means that transitions which are fused will be handled as parts of the fusion group and not as seperate transitions.

Definition 5.2.3
Let a Modular CP-net be given and let W be a weight function.

(i) W is a place flow $\Rightarrow$ W determines a place invariant.

(ii) If no binding elements are dead:
W is a place flow $\Leftrightarrow$ W determines a place invariant.

Proof: The theorem can be proved similarly to Theorem 5.1.3, we just need to consider place groups and transition groups instead of places and transitions.

♦

5.3 How to find Place Invariants of Modular CP-nets

In the examples presented in section 2 and 3, we have shown some compositions of place invariants and place flows, using either place fusion only or transition fusion only.

We use the term "global" weight function for a weight function of the entire modular *CP*-net, while we use the term "local" weight function for weight functions of a single module. In the present section, we state and prove a number of theorems specifying how local place flows and local place invariants are related to global place flows and global place invariants.

We use $S(x)$ to denote the module containing the node x.

Definition 5.3.1
Let $MCPN = (S, PF, TF)$ be a modular *CP*-net.

(i) A set of loyal weight functions $\{W_s\}_{s \in S}$ of the modules is **consistent** iff they have the sue range and assign the same weights to all members of each place group:
$\forall p' \in PG : \forall p_1, p_2 \in p' : W_{S(p_1)}(p_1) = W_{S(p_2)}(p_2)$

(ii) A global weight function W of $MCPN$ **determines** a consistent set of local weight functions $\{W_S\}_{s \in S}$ of the modules:
$\forall p \in P : W_{S(p)}(p) = W([p])$

(iii) A consistent set of loyal weight functions $\{W_s\}_{s \in S}$ of the modules of $MCPN$ **determines** a global weight unction W:
$\forall p' = [p] \in PG : W(p') = W_{S(p)}(p)$.

Note that the construction of (ii) and (iii) will yield valid weight functions and that the construction fulfils: if W^a determines $\{W_s\}_{s \in S}$ and $\{W_s\}_{s \in S}$ determines W^b then $W^a = W^b$.

Definition 5.3.2
Let $MCPN = (S, PF, TF)$ a module *CP*-net and let $\{W_s\}_{s \in S}$ be a consistent set of local weight functions of the modules which determine the global weight function. Then we have:

$\forall s \in S : [W_s \text{ is a place flow of } T_s] \Rightarrow.$
W is a place flow of $MCPN$.

Proof: The theorem follows directly from the observation that all positions of the individual modules are flow preserving, i.e., we know that each member of a transition group is flow preserving. This is a much stronger demand that the transition group being flow presuming as a group. ♦

Note that we cm find plate flows of the total system which cannot be expressed as place flows of the individual modules.

Definition 5.3.3
Let $MCPN = (S, PF, TF)$ a module CP-net without place fusion, and let $\{W_s\}_{s \in S}$ be a consistent set of local weight functions of the modules which determine the global weight function W. Then, we have
$\forall s \in S : [W_s$ determines a place invariant of module $s] \Rightarrow$
W determines a place invariant of $MCPN$.

Proof: From the definition of enabling for Modular CP-nets, we know that a transition group will only be enabled if all transitions of the group are enabled. This means that the set of reachable states for the modular CP-net is covered by the reachable states of the local modules. ⧫

Note that we can find place invariants of the total system which cannot be expressed as place invariants of the individual modules.

Definition 5.3.4
Let $MCPN = (S, PF, TF)$ a module CP-net without place fusion, and let W be a global weight function of $MCPN$ which determines a set of local weight functions $\{W_s\}_{s \in S}$. Then, we have
W is a place flow of $MCPN \Rightarrow$
$\forall s \in S : [W_s$ is a place flow of $T_s]$.

Proof: Since transition fusion is not used we know that all transition groups have exactly one member. This means that each individual transition is flow preserving. ⧫

From definition 5.3.1 and theorem 5.3.4 we know that we can find all place flows of the total system from the place flows of the individual modules. This is an important result which can be applied directly to hierarchical CP-nets since these use hierarchical concepts built on place fusion only.

Definition 5.3.5
Let $MCPN = (S, PF, TF)$ a module CP-net and let W be a global weight function of $MCPN$ which determines a set of local weight functions $\{W_s\}_{s \in S}$. Then, we have:

W is a place flow of the Modular CP-net $\Rightarrow$
$\forall s \in S : [W_s$ is a place flow of $IT_s] \wedge \forall tf \in TF : [tf$ is flow preserving for $W]$.

Proof: In this proof, it is sufficient to establish a correspondence between the transition groups and the union of the set of internal transitions and the set of fusion sets. If a transition group contains exactly one transition it corresponds to an internal transition and otherwise it corresponds to a transition fusion set. ♦

Theorem 5.3.5 is the main result presented in this paper. All place flows of a modular CP-net can be determined from consistent sets of weight functions which are place flows of the internal transitions IT and are place preserved by all transition fusion sets. The theorem has been shown for place fusion only in [NV85]. When the two results are compared, please note that place flows in our terminology comespond to place invariants in [NV85].

To illustrate how we envision the use of the results from this section, we describe how a place flow could be found for a modular CP-net. We will assume that a modular CP-net has mainly internal, i.e., non-fused, transitions and places, and the main work is to check that the internal transitions are flow preserving. When you perform place invariant analysis you can be interested either in all possible flows of the system or in a few but descriptive place flows. If you want to calculate all possible place flows, theorem 5.3.5 allows you to do this in a modular way. Calculate all flows of the internal transitions and combine the ones which are consistent, and finally check the transition fusion sets.

If you want to use an interactive process where you gradually add weights of places the result of the theorem is also attractive. You start with a single module and specify weights that are flow preserved by all internal transitions. After this, you can use the weights of the module to restrict the rest of the weight functions. You can use both place fusion sets and transition fusion sets. We know that the weight functions must be consistent and this means that all places of a place fusion set must have the same weight. We also know that the transition fusion sets must be flow preserving and if only one of the surrounding places needs a weight that can be calculated directly from the known weights. After these restrictions have been applied the internal transitions of the next module can be checked. In the end, it is necessary to check that all transition fusion sets are flow preserving.

5.4 An example of a Modular Approach to Place Invariants

We have tried to use the results from section 5.3 to find place flows of the hierarchical *CP*-net described in [CJ91]. This is a model describing a detailed design of the Network Management System of the RcPAX X.25 wide area network. It consists of 30 pages, many of these having up to 7 instances due to the reuse of pages. The modular approach made it easy to find the place invariants needed in the proof of properties which were local to few pages. The modular approach also made it possible to compose place flows of the individual page instances into place flows of the total system. An example of a property which could be proved directly by means of a place invariant and which involved many pages was the preservation of packages in the system. The handling of packages was relatively complex and involved grouping packages into larger logical units. By adding extra places which would keep a log of the information passed on the network, it was possible to investigate how messages could be lost and check that the information which was re-established either matched the original message, or the originating sender would be notified. The work on modular place invariants was performed after the design of the model was done, and not as an integrated part of the modelling process. It should also be noted that the work on the examples was done by hand. Tool support will be necessary if place invariants should be used as part of the development of large descriptions. It is our impression that a similar approach could easily be applied to other hierarchical *CP*-net models, e.g., the ISDN Basic Rate Interface described in [HP91].

We have not investigated how a modular approach can be used for large systems related by means of transition fusion since we have no models of this nature at our disposal. From our own experiments with small systems it seems to be possible to use a modular approach for larger ones too.

In this section, we have formally defined place invariants for *CP*-nets. Then, we have extended this definition to modular *CP*-nets in such a way that place invariants of a module *CP*-net correspond to place invariants of the equivalent *CP*-net. We have also shown how place flows of the modular *CP*-net can be determined from place flows of the individual modules. It is possible to obtain the dual results of place invariants for transition invariants, but we omit this.

6 Conclusion

In this paper, we have presented a framework for modular analysis of *CP*-nets. We have considered sets of individual *CP*-nets related by transition fusion and by place fusion. Transition fusion can be used to model synchronous actions, while place fusion can be used to model shared data. Modular *CP*-nets form a simple but yet very general framework to discuss analysis of structured net models.

We also introduced analysis of modular *CP*-nets by means of place flows. It allows us to determine place flows of the modular *CP*-net from the individual modules, only transition fusion needs to be checked globally. This means that it is not necessary to recompute all place flows when a single module is modified.

The intention of modular *CP*-nets is to provide a theoretical framework to discuss the analysis of structured net models. The formal relation between pages of a Hierarchical *CP*-net [Jen91] can be represented as modular *CP*-nets using place fusion only. The communication primitives of Channel *CP*-nets [CD92] can be represented as modular *CP*-nets using transition fusion only.

It would be possible to use a modular approach for other Petri net models. As an example, some kinds of Petri nets with algebraic specifications, see e.g. [DHP91], [Rei91] and [BDM91], use transition fusion. From [BPR91], we know that OBJSA nets ([BDM91]) can be translated into algebraic nets schemes. And it would be straightforward to apply the modular approach to other kinds of Petri Nets in order to obtain modular algebraic nets schemes, with similar results as modular *CP*-nets.

Another approach to modularity and nets, which consists in automatically decomposing a flat net into modules, is presented for Place/Transition nets in [FJP91]. As the decomposition method only relies on the net structure, it is also valid for *CP*-nets.

A work to construct occurrence graphs of modular *CP*-nets is ongoing, see [HJJJ86] and [Jen91] for information about occurrence graphs of *CP*-nets. But several difficult problems arise, similar to those pointed out in [Val90] and [FP91]. Since the state spaces of systems tend to grow exponentially in the number of processes of the system ([Val90]), a modular approach to occurrence graphs would be of very large interest.

Acknowledgements

Many thanks to Kurt Jensen for the fruitful discussions we held on the subject. We would also like to acknowledge Gérard Berthelot, Niels Damgaard Hansen, Mogens Nielsen and the anonymous referees whose comments helped us to improve our paper.

References

[BDM91] E. Battiston, F. De Cindio, G. Mauri: **OBJSA nets systems: a class of high-level nets having objects as domains.** In: G. Rozenberg (ed.): Advances in Petri Nets 1988. Lecture Notes in Computer Science, vol 340. Springer-Verlag, 1988, pp. 20–43. Also in [JR91], pp. 189–212.

[BPR91] E. Battiston, L. Petrucci, L. Rapanotti: **Establishing a relationship between OBJSA nets systems and algebraic nets schemes.** DEMON Esprit BRA n° 3148, Technical Report 185, 1991.

[CD92] S. Christensen, N. Damgaard Hansen: **Coloured Petri nets extended with channeIs for synchronous communication.** Daimi PB-390, ISSN 0105-8517, April 1992.

[Che91] G. Chehaibar: **Use of reentrant nets in Modular analysis of coloured Petri nets.** In: G. Rozenberg (ed.): Advances in Petri Nets 1991. Lecture Notes in Computer Science, vol 524. Springer-Verlag, 1991, pp. 58–77. Also in [JR91], pp. 596–617.

[CJ91] S. Christensen, L. O. Jepsen: **Modelling and simulation of a network management system using hierarchical coloured Petri nets.** In Erik Mosekilde (ed.): Proceedings of the 1991 European Simulation Multiconference. ISBN 0-911801-92-8, pp. 47–52. An extended version available as: Daimi PB-349, ISSN 0105-8517, April 1991.

[DHP91] C, Dimitrovici, U. Hummert, L. Petrucci: **Semantics, composition and net properties of algebraic high-level nets.** In: G. Rozenberg (ed.): Advances in Petri Nets 1991. Lecture Notes in Computer Science, vol 524. Sponger-Verlag, 1991, pp. 93–117.

[FJP91] A. Finkel, C. Johnen, L. Petrucci: **Decomposition of Petri nets for parallel analysis.** CEDRIC Research Repour November 1991.

[FP91] A. Finkel, L. Petrucci: **Avoiding state explosion by composition of minimal covering graphs.** Proceedings of the 3rd Computer-Aided Verifiation Workshop, Ålborg, Denmark, July 1991, pp. 224–237. To appear in Lecture Notes in Computer Science.

[HJJJ86] P. Huber, A. M. Jensen, K. Jensen, L. O. Jepsen: **Reachability trees for high-level Petri nets.** Theoretical Computer Science n°45, 1986, pp. 261–292. Also in [JR91], pp. 319–350.

[HJS90] P. Hubert K. Jensen and R. M. Shapiro: **Hierarchies in coloured Petri nets.** In: G. Rozenberg (ed.): Advances in Petri Nets 1990. Lecture Notes in Computer Science, vol 383. Springer-Verlag, 1990, pp. 342–416. Also in [JR91], pp. 215–243.

[HP91] P. Hubert V. O. Pinci: **A formal, executable specification of the ISDN basic rate interface.** Proceedings of the 12th International Conference on Application and Theory of Petri Nets, Aarhus, Denmark, 1991, pp. 1–21.

[Jen81] K. Jensen: **Coloured Petri nets and the invariant method.** Theoretical Computer Science n°14, 1981, pp. 317–336.

[Jen86] K. Jensen: **Coloured Petri nets.** In: G. Rozenberg (ed.): Advances in Petri Nets 1986, Part I. Lecture Notes in Computer Science, vol 254. Springer-Verlag, 1986, pp. 248–299.

[Jen91] K. Jensen: **Coloured Petri nets: A high level language for system design and analysis.** In: G. Rozenberg (ed.): Advances in Petri Nets 1990. Lecture Notes in Computer Science, vol 383. Springer-Verlag, 1990, pp. 342–416. Also in [JR91], pp. 44–119.

[Jen92] K. Jensen: **Coloured Petri nets. Basic concepts, analysis methods and practical use. Volume 1: Basic concepts.** To appear in EATCS monographs on Theoretical Computer Science, Springer-Verlag 1992.

[JR91] K. Jensen and G. Rozenberg (eds.): **High-level Petri nets: theory and application.** Spinger-Verlag 1991. ISBN 3-540-54125-X/0-387-54125-X.

[NV85] Y. Narahari, N. Viswanadham: **On the invariants of coloured Petri nets.** In: G. Goos and J. Hartmanis (eds.): Advances in Petri Nets 1985. Lecture Notes in Computer Sciences vol 222. Springer-Verlag, 1985, pp. 330–341.

[Rei91] W. Reisig: **Petri nets and algebraic specifications.** Theoretical Computer Science n°80, 1991, pp. 1–34. Also in [JR91] pp. 137–170.

[Val90] A. Valmari: **Compositional state space generation.** Proceedings of the 11th iterations Conference on Application and Theory of Petri Nets, Paris, France, June 1990, pp. 43–62.