Coloured Petri Nets Extended with Channels for Synchronous Communication

Søren Christensen
Niels Damgaard Hansen
Computer Science Department, Aarhus University
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C, Denmark
Phone: +45 89 42 31 88
Telefax: +45 89 42 32 55
E-mail: schristensen@daimi.aau.dk, ndh@daimi.aau.dk

Abstract

This paper shows how Coloured Petri Nets (CP-nets) can be extended to support synchronous communication. We introduce coloured communication channels through which transitions are allowed to communicate complex values. Small examples show how channel communication is convenient for creating compact and comprehensive models.

The concepts introduced in this paper originate from the practical use of Petri nets for modelling, and they are formally defined in such a way that they preserve the basic properties of CP-nets. We show how a CP-net with channels can be transformed into a behaviourally equivalent CP-net. This allows us to deduce properties of CP-nets with channels from well-known properties of CP-nets. As an example, we extend the concept of place invariants to cope with CP-nets with channels and show how place invariants can be found. This is done without transforming the CP-nets with channels into their equivalent CP-nets.

The reader is assumed to be familiar with the notion of CP-nets.

Keywords:

Coloured Petri nets, synchronous communication, channels, modular specifications, re-usable models, invariant analysis.

The work presented in this paper has been supported by a grant from the Danish Research Programme for Informatics—grant number 5.26.18.19.

Introduction

During our involvement in modelling projects using hierarchical CP-nets [HJS90], [Jen91], [Jen92], it has become clear that it would be valuable to include constructs making it easy to model synchronisation and synchronous communication. Without such constructs it is necessary to model explicitly the synchronous communication through additional places and transitions often resulting in a complex net structure. This means that the modeller has to devote much attention to the model, instead of focusing on the problem being modelled. We propose to extend CP-nets to support communication through channels. The concept of channels is influenced by CCS, CSP and communication constructs found in high level programming languages, e.g., [Hoa85] and [Mil89]. For the sake of simplicity we show how to extend CP-nets and not hierarchical CP-nets. It should, however, be obvious that the extension can be generalised to hierarchical nets.

Other Petri net models have used the notation of communication channels, e.g., [HT91], but the concept has, to the best of our knowledge, never been formally defined before and never fully integrated into the Petri net framework.

The paper is organised as follows: first, the new constructs are informally introduced. Then we show how CP-nets with channels can be formally defined and transformed into behaviourally equivalent CP-nets. In section 3 we show how analysis methods—especially place invariants—can be extended to cope with CP-nets with channels without having to transform the models into the equivalent CP-nets. Finally, we present a number of small examples illustrating the convenience of CP-nets with channels for modelling.

In this paper we have left out the technical details of some of the proofs. The detailed proofs can be found in [CD92].

1. Informal Introduction to CP-nets with Channels

In this section we introduce CP-nets with channels. We informally describe the new concepts by means of small examples. These examples illustrate how channels may be used for widely different purposes.

Extending CP-nets with channels allow transitions to communicate through named and coloured communication channels. Transitions, which use channel communication, are called communication transitions and for each channel they are divided into !?-transitions and ?!-transitions. A communication between two transitions is only possible if one of the transitions is a !?-transition and the other is a ?!-transition—and they use the same channel. No direction of communication is intended, this is the reason for using !? and ?! instead of just ! and ? used in CSP. !? and ?! is merely used to divide the transitions into two groups and it would make no difference if all !? were exchanged by ?!.

An example of a CP-net with channels is shown in Fig. 1.1. The inscriptions next to the communication transitions Send and Receive are called communication expressions specifying the channel and the actual value communicated through the channel.

We explicitly declare the colour set, i.e., the type, of the communication channel. This makes static type checking of the communication expressions possible, since we, based on the net-inscriptions, are able to determine if the communication expressions evaluate to values of the same type as the associated communication channel.

Intuitively, a communication through a channel ch is enabled if and only if there exist two communication transitions t_1 and t_2 with communication expressions $expr_1$ and $expr_2$ and two bindings b_1 and b_2 such that:

- transition t_1 is enabled for the binding b_1 and t_2 is enabled for the binding b_2, i.e., there are sufficient tokens of the correct colours on the input places,
- t_1 has a communication expression of the form $expr_1$!?ch,
- t_2 has a communication expression of the form $expr_2$?!ch,
- $expr_1<b_1> = expr_2<b_2>$, i.e., $expr_1$ and $expr_2$ evaluate to the same value when they are evaluated in the bindings for which the two transitions occur.

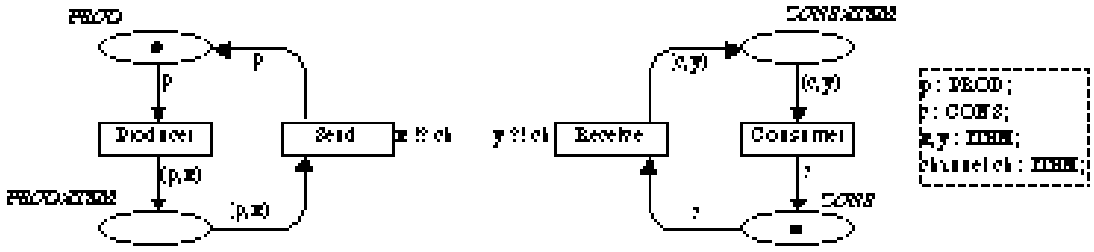

Figure 1.1: CP-net with two communication transitions using a channel

Please note that although most of the examples in this paper show transitions communicating on one channel only, our formalism are able handle transitions communicating on multiple channels.

The semantics of a CP-net with channels may be illustrated by constructing the behaviourally equivalent CP-net. Fig. 1.2 shows the behaviourally equivalent CP-net of the CP-net with channels shown in Fig 1.1. Intuitively the equivalent CP-net is constructed by merging the !?-transition and the ?!-transition so that the arcs of the merged transition are the union of the arcs of the original transitions. The guard of the merged transition is formed by the conjunction of the original guards and an expression stating that the communication expressions must evaluate to the same value. Section 2.3 describes this transformation in more detail.

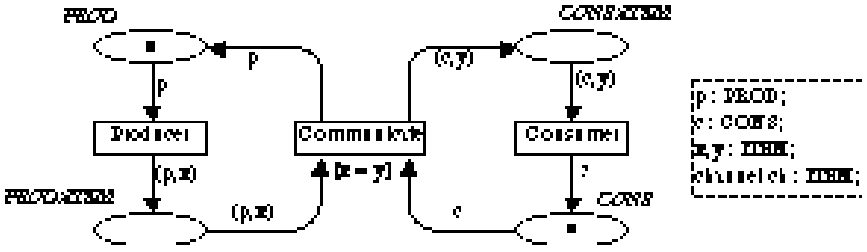

Figure 1.2: Equivalent CP-net of the CP-net with channels in Fig. 1.1

It should be noted that information during a communication may be passed in both directions, i.e., from a !?-transition to a ?!-transition, or vice versa. In the example above the variable x appears in the input arc expression of Send, while the variable y appears in the output arc expression of Receive. This means that the colour of the input token of Send determines the colour of the output token of Receive, and thus information is sent (through the channel) from Send to Receive. In the general case, two communication transitions may have communication expressions specifying bi-directional exchange of information. Throughout this paper we have chosen to use the !?-transition to denote the communication transition determining the value communicated through the associated

channel. This convention can of course not always be used, e.g., when the communication is bi-directional.

If we extend the CP-net of Fig. 1.1 to consist of two producers and two consumers we get the CP-net with channels presented in Fig. 1.3. Often, we would fold the producers and consumers into a single structure and distinguish the processes by means of the associated colours—but let us assume that all four processes have internal structures that make it difficult to make this folding. All processes communicate using the channel called ch. In this way each producer is able to send to any of the consumers ready to receive data.

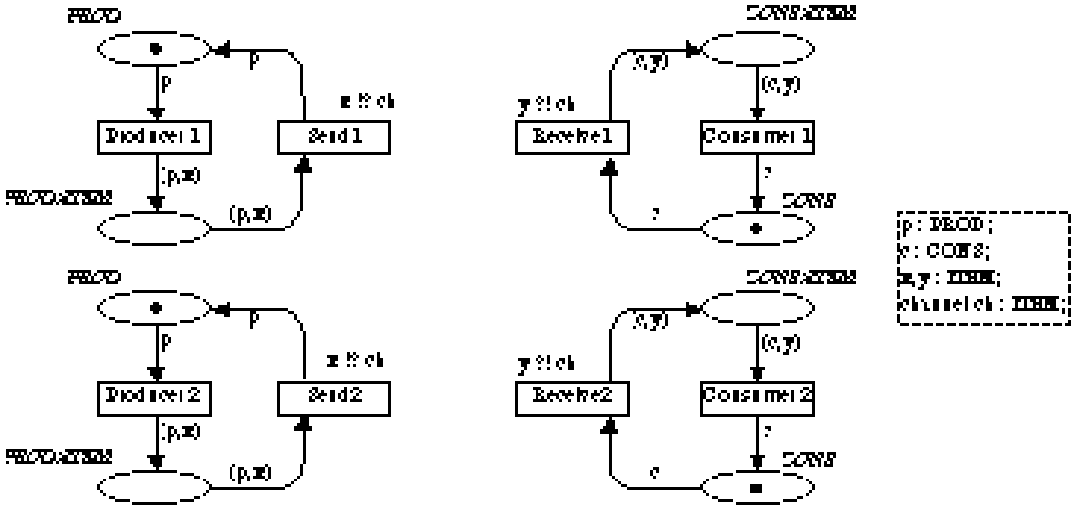

Figure 1.3: Producer-consumer modelled using CP-nets with channels

Fig. 1.4 illustrates that the behaviourally equivalent CP-net becomes rather complex. If we have n !?-transition and m ?!-transition for a given channel we will need n m transitions to model the same patterns of communication. This observation illustrates a reason why the introduction of channels may result in a dramatic simplification of the net structure. When applied to nodes in a single net, e.g., in Fig. 1.3, channel communication is mainly a drawing convenience allowing the user to avoid too many crossing arcs. Fig. 1.4 illustrates that this is a valuable quality. Channel communication becomes a strong description primitive of its own right especially when it is combined with structured net descriptions such as hierarchical CP-nets, where a model is described as a set of related subnets, e.g., drawn on separate pages.

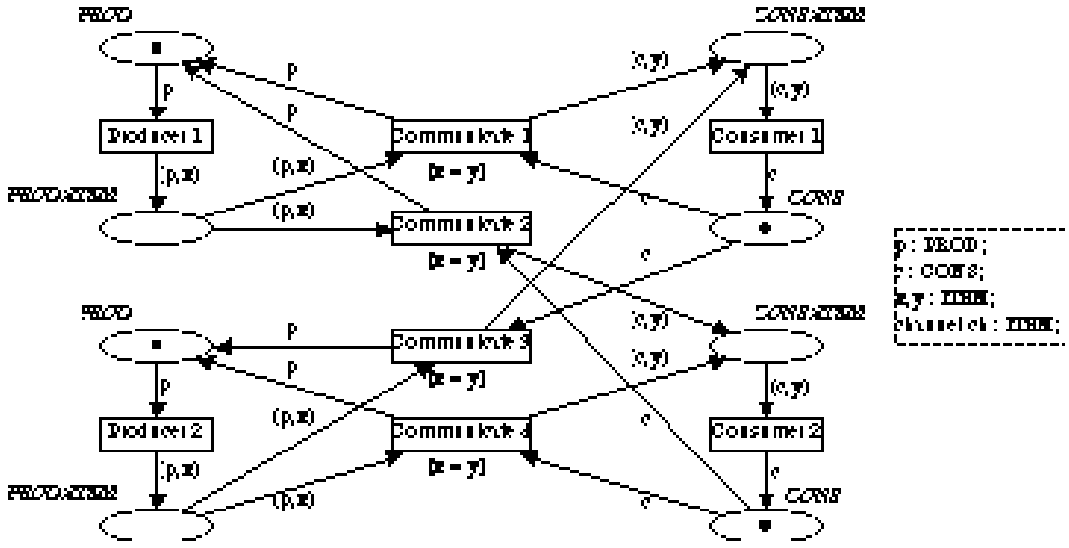

Figure 1.4: Equivalent CP-net of the CP-net with channels in Fig. 1.3

The example of Fig. 1.3 also illustrates one of the differences between channel communication and transition fusion described in [HJS90] and [CP92]. When we compare the concept of channel communication to the concept of transition fusion we find two major differences: first, the bindings of transitions involved in a channel communication are independent except for the restrictions expressed in the communication expressions. This is in contrast to transition fusion where the transitions share a common binding. Secondly, the occurrence of a transition fusion set always involves all members of the fusion set. For a CP-net with channels an occurrence involves pairs of !?-transitions and ?!-transitions. This means that some communication transitions using a given channel may occur without involving the other communication transitions using the same channel. So, if we would model channel communication by means of transition fusion, we would need to create a transition fusion set for each possible pattern of communication, i.e., to express the communication of the example of Fig. 1.3 we would need to create 4 transition fusion sets. Although this means that the behaviour of a CP-net with channels can be expressed by means of transition fusion, this is often not feasible from a modelling point of view. Transition fusion in [CP92] is defined to facilitate the discussion of modular analysis of CP-nets while CP-nets with channels are defined to ease modelling. The main advantage of CP-nets with channels is the possibility to create and analyse independent and re-usable sub-models.

The model shown in Fig. 1.3 is structured in such a way that the producers and the consumers are described in separate nets, while the channel defines the interface between the otherwise independent sub-models. It is possible, e.g., to add another consumer or change the internal description of a consumer independently of the other descriptions—as long as the channel specifying the interface between the descriptions is not changed.

Until now, we have only considered data communication. It is, however, also easy to model communication without exchange of data, i.e., mere synchronisation. This can be done by using an elementary channel colour E = {e} containing only one possible value. It is analogous to the way in which we handle tokens carrying no information in their colour.

In the last example of this section we use CP-nets with channels to model an abstract data type. We use the term abstract data type to stress that the internal representation of the information is accessed through a fixed interface. Abstract data types may in a CP-net be expressed as a colour set having a number of operations. It can, however, also be convenient to model data types through small subnets allowing us to reason about the net structure, e.g., to use invariants.

In Fig. 1.5, we describe a queue and its operations. This queue could for instance be inserted between the producers and the consumers describing that items from the producers are buffered before arriving at the consumers.

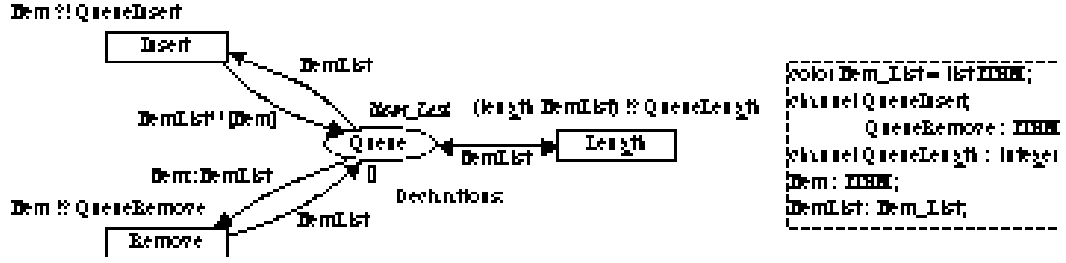

Figure 1.5: CP-net with channels describing a queue with operations

We have introduced a channel for each of the basic operations: Insert, Remove and Length. The queue itself is a token on the Queue place consisting of a list with the buffered elements. Initially the queue is empty as indicated by []. Elements are inserted at the end of the list using ^^ for list concatenation. The list constructor :: is used to split the list into its head element and tail.

The communication transition modelling an operation is enabled by communicating through the associated channel, e.g., the Insert transition is enabled by communicating an item through the Insert channel.

An alternative way of implementing the queue would be to use only one channel and communicate both operation and data through the channel to let the queue implementation distinguish the operations. Similar to this we could implement multiple queues without duplicating the net structure simply by extending the channel communication to contain information on the queue we want to access.

If we want to add a queue between the producers and the consumers we just have to change their use of channels, see Fig 1.6.

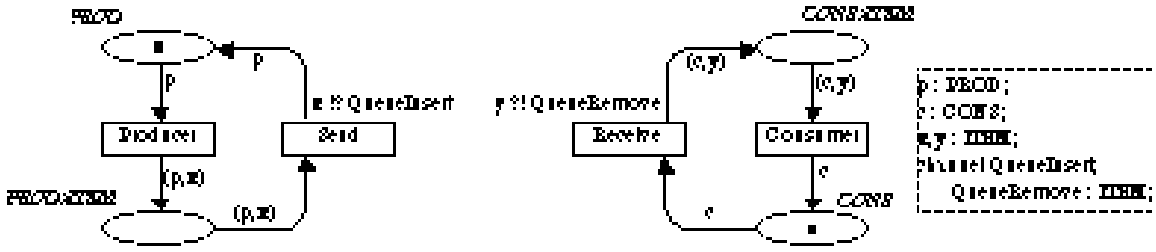

Figure 1.6: Producer and consumer using a queue

The producer uses the QueueInsert channel to insert an element into the queue, whereas the consumer uses the QueueRemove channel to obtain an element from the queue. In this example we do not use the Length operation.

2. Formal Definition of CP-nets with Channels

In this section we formalise how channel communication can be added to CP-nets. The notation used in this chapter is consistent with the one used in [Jen91] and [Jen92]. First we recall the definition and notation for multi-sets.

A **multi-set** m, over a non-empty set S, is a function m $[S \rightarrow \mathbb{N}]$. The non-negative integer m(s) $\mathbb{N}$ is the number of appearances of the element s in the multi-set m. We usually represent the multi-set m by a formal sum:

$$\sum_{s \in S} m(s)`s$$

By S_{MS} we denote the set of all multi-sets over S. The non-negative integers {m(s)½s S} are called the **coefficients** of the multi-set m, and the number of appearances of s, m(s), is called the coefficient of s. An element s S is said to **belong** to the multi-set m iff m(s) ≠ 0 and we then write s m.

2.1 Definition of CP-nets with Channels

We recall the definition of non-hierarchical CP-nets:

Definition 2.1 ([Jen92], Def. 2.5)
A **CP-net** is a tuple CPN = (Σ, 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 ∩ T = P ∩ A = T ∩ A = Ø.
(v) N is a **node** function. It is defined from A into P × T T × 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 boolean expressions such that:
  • t T: [Type(G(t)) = 𝔹 Type(Var(G(t))) Σ].
(viii) E is an **arc expression** function. It is defined from A into expressions such that:
  • a A: [Type(E(a)) = C(p(a))$_{MS}$ Type(Var(E(a))) Σ]
  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:
  • p P: [Type(I(p)) = C(p)$_{MS}$].

A detailed explanation of the requirements of this definition can be found in [Jen91] and [Jen92]. For an arc, a, we use p(a), t(a), s(a) and d(a) to denote the place, transition, source and destination respectively. For a node x we use A(x) to denote the set of arcs connected to x. We furthermore use 𝔹 to denote the boolean type containing the elements {false, true} and having the standard operations from propositional logic.

We use the set Δ = {!?, ?!} to denote the two relations a communication transition can have to a channel. Moreover, we use Var(expr) to denote the variables appearing in an expression expr and we use Type(v) and Type(expr) to denote the type of a variable v and an expression expr, respectively. For more information about the use of types, expressions and variables in CP-nets see [Jen91] or [Jen92]. We are now ready to give a formal definition of CP-nets with channels. A short explanation is given below the definition. It is recommended to read this explanation in parallel with the definition.

Definition 2.2
A CP-net with channels is a tuple CCPN = (CPN, CS) satisfying the requirements below:

(i) CPN is a non-hierarchical CP-net (Σ, P, T, A, N, C, G, E, I).
(ii) CS is a **channel specification** (CH, CT, CE)
(iii) CH is a finite set of **channels** such that:
  • (P T A) ∩ CH = Ø.
(iv) CT is a **channel type** function. It is defined from CH into Σ.
(v) CE is a **communication expression** function. It is defined from T into finite sets of communication expressions. All elements of CE(t) are triples (expr, #, ch) where expr is an expression, # Δ and ch CH such that:
  • t T: (expr,#,ch) CE(t):
    [Type(expr) = CT(ch) Type(Var(expr)) Σ].

(i) CPN is a CP-net, see Def. 2.1.

(ii) The channel specification is a triple containing a set of channels, a channel type function and a channel expression function.

(iii) CH is a finite set of channels, which can be distinguished from the net elements of the CP-net. In Fig. 1.1 we have: CH = {ch}.

(iv) The channel type specifies the kind of information that can be passed through a given channel. In Fig. 1.1 we have: CT(ch) = ITEMS.

(v) The expression of a communication expression must have a type identical to the type of the channel. Moreover, all variables must be of a known type. In Fig. 1.1 we have: CE(Producer) = {}, CE(Consumer) = {}, CE(Send) = {(x, !?, ch)} and CE(Receive) = {(y, ?!, ch)}. We could generalise the communication expression function to return multi-sets of communication expressions, but since it is not important for the results shown in this paper we have chosen to use plain sets.

For $t \in T$, $\# \in \Delta$ and $ch \in CH$ we use Expr(t,#,ch) to denote the set of communication expressions connecting t and ch in direction #:

$$Expr(t,\#,ch) = \{expr' \mid (expr',\#,ch) \in CE(t)\}.$$

We generalise the expression function E to handle channel expressions. E(t,#,ch) denotes the multi-set sum of all expressions in Expr(t,#,ch):

$$E(t,\#,ch) = \sum_{expr \in Expr(t,\#,ch)} 1`expr.$$

The multi-set sum in E is well-defined since all expressions of a given channel evaluate to value of the channel type.

2.2 Behaviour of CP-nets with Channels

Having defined the static structure of CP-nets with channels we are now ready to consider their behaviour. Adding communication to a given transition constrains its enabling, but does not change the effect of an occurrence.

First we re-define the set of variables for a transition so that it also includes the variables of the communication expressions:

$$\forall t \in T: Var(t) = \{v \mid v \in Var(G(t)) \vee \exists a \in A(t): v \in Var(E(a)) \vee \exists (expr,\#,ch) \in CE(t): v \in Var(expr)\}.$$

Our definitions of bindings, B(t), token elements, TE, binding elements, BE, markings, $\mathbb{M}$, and steps, $\mathbb{Y}$, are identical to the definitions given in [Jen92], which except for technical details are identical to those of [Jen91]. However, the enabling rule of CP-nets must be extended to take into account the communication expressions.

Definition 2.3

A step $Y \in \mathbb{Y}$ is **enabled** in a marking $M \in \mathbb{M}$ iff the following properties are satisfied:

(i) $$\forall p \in P: \sum_{(t,b) \in Y} E(p,t)<b> \leq M(p).$$

(ii) $$\forall ch \in CH: \sum_{(t,b) \in Y} E(t,!?,ch)<b> = \sum_{(t,b) \in Y} E(t,?!,ch)<b>.$$

If Y fulfils (ii) it is said to be **communication enabled**.

(i) This is the enabling rule of CP-nets, See [Jen91] Def. 2.6 or [Jen92] Def 2.8.

(ii) For each channel we demand that the multi-set of values obtained by evaluating the !?-expressions must match the multi-set of values obtained by evaluating the ?!-expressions. This means that a step is enabled iff all binding elements are enabled and the communication can take place.

Our definitions of occurrence, occurrence sequence and reachability are identical to those of [Jen91] and [Jen92].

2.3 Behaviourally Equivalent CP-net

In this section we show, that although adding channels to CP-nets increases the possibility for creating compact and comprehensive models it does not increase the computational power. We show that a large class of CP-nets with channels, which we call well-formed, can be transformed into behaviourally equivalent CP-nets. By a behaviourally equivalent CP-net, we mean a CP-net without channels behaving like the original CP-net with channels—that is, there is a one to one correspondence between markings and enabled steps of the two models (see Theorem 2.5). The existence of an equivalent CP-net is extremely useful, because it tells us how to generalise the basic concepts and the analysis methods of CP-nets to CP-nets with channels. We simply define these concepts in such a way that a CP-net with channels has a given property iff the equivalent CP-net has the corresponding property. It is important to understand that we never make the transformation for a particular CP-net with channels. When we describe a system we directly use CP-nets with channels without constructing the equivalent CP-net. Similarly, we directly analyse a CP-net with channels without constructing the equivalent CP-net.

In Def. 2.3 we defined the enabling rule of CP-nets with channels. This definition gives an operational semantics that is easy to understand and use. In this section we show how to transform the restrictions imposed by the additional enabling restriction into a structural property of the resulting CP-net. The formal proof involves a number of definitions and mathematical proofs. In this paper we have chosen to give the intuition behind the transformation and only the most central definitions and proofs. The formal definitions and proofs are found in [CD92].

The basic idea behind the transformation of a CP-net with channels to an equivalent CP-net is illustrated in Fig. 1.2 and Fig. 1.4: all transitions involved in a channel communication are merged together so that the arcs of the merged transition are the union of the arcs of the original transitions and the guard of the merged transition is formed by the conjunction of the original guards and an expression stating that the communication expressions must evaluate to the same value.

Since the bindings of the communication transitions involved in a channel communication are independent except for the restrictions expressed in the communication expressions we need to make sure that set of variables of the involved communication transitions are disjoint before we merge the transitions. Therefore, for each transition, t, we replace each variable v Var(t) with a new variable v_t, of the same type as v. We assume that the names of the new variables are different.

Since the transformation is based only on the net structure and does not take the inscriptions into account, the transformation obviously fails to be finite if the communication contains cycles. A simplified example of a net with a cyclic communication is shown in Fig. 2.1. The t1 transition initiates the communication by sending an integer value on channel ch1. If the value is negative t2 just receives the value. Otherwise t3 sends the

decremented value on ch2. t4 just retransmits on ch1 the value received on ch2. The transformation of this net fails to be finite because it is not possible from the net structure alone to determine how many instances of t3 and t4 should be merged. It depends on the actual value of the variable v, e.g., if v=3 the transformation would include one instance of t1 and t2, but 3 instances of t3 and t4.

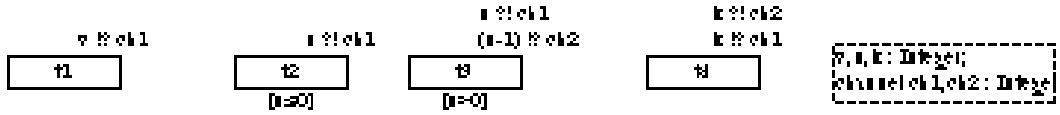

Figure 2.1: Cyclic channel communication

In the rest of this paper we only consider what we call well-formed CP-nets with channels. We say that a CP-net with channels is **well-formed** iff the channel communication has no directed cycles. Well-formed CP-nets with channels have behaviourally equivalent CP-nets. It should be noted that not well-formed CP-nets with channels may have a reasonable meaning—although they are forbidden in order to be able to transform CP-nets with channels to CP-nets.

In the definition of the equivalent CP-net we need to define which transitions are merged together. We define the term **transition groups**. Intuitively a set of transitions form a transition group iff they fulfil:

- if the transition is an ordinary transition it constitutes a transition group of its own,
- for each channel involved in the communication between transitions in the set, we must have exactly one !? and one ?!-transition,
- the transition group is minimal. That is, a transition group contains only one ordinary transition or the communicating transitions involved in one specific communication. Since a communication transition may communicate on several channels a transition group may include transitions communicating on a number of channels.

In Fig. 1.3 we have eight communication groups: one for each of the ordinary transitions, i.e., the producer and consumer transitions and four groups consisting of the possible communication paths, i.e. {Send1, Receive1}, {Send1, Receive2}, {Send2, Receive1} and {Send2, Receive2}.

For each transition group, we are now able to define the conditions that communication enables the transitions. The **communication guard** function CG is defined from the set of transition groups, TG, into boolean expressions:

$$T'' \in TG: CG(T'') = \bigwedge_{ch \in CH} \left[E(T'',!?,ch) = E(T'',?!,ch)\right].$$

Having this notion of transition groups and communication guard, we are now able to define the transformation from CP-nets with channels into a behaviourally equivalent CP-nets:

Definition 2.4 (Def. 2.8 in [CD92])
Let a CP-net with channels CCPN = (CPN, CS) be given, where CPN = (Σ, P, T, A, N, C, G, E, I) and CS = (CH, CT, CE). Then we define the **equivalent CP-net** to be CPN* = (Σ*, P*, T*, A*, N*, C*, G*, E*, I*) where:
(i) $\Sigma^* = \Sigma$.
(ii) $P^* = P$.

(iii) $T^* = TG$.
(iv) $A^* = \{(a,t^*) \in A \times T^* \mid t(a) = t^*\}$.
(v) $a^* = (a, t^*) \in A^*$:
$[s(a) \in P \Rightarrow N^*(a^*) = (p(a),t^*)$
$s(a) \in T \Rightarrow N^*(a^*) = (t^*,p(a))]$.
(vi) $C^* = C$.
(vii) $t^* \in T^*$: $G^*(t^*) = \bigwedge_{t \in t^*} G(t) \wedge CG(t^*)$.
(viii) $a^* = (a,t^*) \in A^*$: $E^*(a^*) = E(a)$.
(ix) $I^* = I$.

(i) The set of colour sets is unchanged.
(ii) The set of places is unchanged.
(iii) We have a transition for each of the transition groups. We recall that transitions, which do not communicate, are represented as singleton transition groups, i.e., by communication groups containing only a single transition.
(iv) The arcs are duplicated such that for an arc, a, we get a copy of a for each communication group having the transition of a as a member.
(v) The node function is changed to match (iv).
(vi) The colour function is unchanged.
(vii) The guard consists of two independent parts: the conjunction of the guards in the original transitions and the communication guard defined above.
(viii) The arc expression function is changed to match (iv).
(ix) The initialization expression function is unchanged.

The equivalent CP-net has a set of steps that is different from the CP-net with channels. This is because each binding element in the equivalent CP-net by definition fulfils the guard. Thus we only have a binding element when the involved communication expressions have matching values. A similar property is not satisfied for the CP-net with channels, where bindings of the communication transitions may be defined in such a way that the values of the communication expressions do not match. A step with such bindings will of course not be enabled—but the binding elements do exist. All concepts with a star refer to CPN*, while those without refer to CCPN.

Theorem 2.5 (Theorem 2.9 in [CD92])
Let CCPN be a CP-net with channels 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) There exists a bijective function, ϕ, which maps communication enabled steps of CCPN onto steps of CPN*.
(iii) $M_1,M_2 \in \mathbb{M} \wedge Y \in \mathbb{Y}$:
$M_1 [Y\rangle_{CCPN} M_2 \Leftrightarrow M_1 [\phi(Y)\rangle_{CPN^*} M_2$

Proof: The proof (shown in [CD92]) can be derived from the previous definitions.

3. Analysis of CP-nets with Channels

It is important that our extension do not invalidate the analysis methods already known for CP-nets. In practice, it is not sufficient to be able to map a CP-net with channels into

a behaviourally equivalent CP-net. We must also be able to perform analysis directly on the CP-nets with channels. The present section discusses how this can be done. We consider simulation, occurrence graphs and place invariants.

3.1 Simulation and Occurrence Graph Analysis

The main problem of computer support for simulation of CP-nets with channels is to find a good way to represent the channel communication to the user. Especially the representation of locally enabled transitions that are restricted by their communications, can be difficult to present. However, if a simulator can handle CP-nets we see no major conceptual problems in extending it to handle CP-nets with channels.

In occurrence graphs we usually only consider steps containing a single binding element, i.e., a single transition, with a single binding. For CP-nets with channels this is not sufficient, because we need at least two binding elements to have a communication. Thus we must instead consider steps corresponding to transition groups in the equivalent CP-net. However, again we see no conceptual problems.

3.2 Place Invariants

In this section we show how the concepts of place invariant and place flow can be extended to CP-nets with channels. Place invariants can be used in the proofs of properties of a CP-net, e.g., absence of dead markings. In this paper, we focus on the concepts of place invariants and place flows more than on the use of invariants and flows in the proof of properties of CP-nets. For an introduction to invariants for CP-nets see [Jen81], [Jen86], [Jen91] and [Jen92].

3.2.1 Formal Definition of Place Invariants for CP-nets

In the following, we formally define the concepts of place invariants and place flows for CP-nets. Before doing this it is necessary to define the notion of weighted set.

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 is allowed. The set of all weighted sets over S is denoted by S_{WS}. Weighted sets have properties that 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 3.1

Let CPN = (Σ, P, T, A, N, C, G, E, I) be a CP-net. We then define:

(i) W is a **place weight** function, with range $R \subseteq \Sigma$:

$$\forall p \in P: W(p) \in [C(p)_{WS} \rightarrow R_{WS}]_L.$$

(ii) For each marking M the place weight function W determines $W(M) \in R_{WS}$:

$$\forall M \in \mathbb{M}: W(M) = \sum_{p \in P} W(p)(M(p)).$$

(iii) The place weight function W determines the **place invariant** $W(M) = W(M_0)$ iff the weighted marking is constant for all reachable markings:
$M \in [M_0\rangle: W(M) = W(M_0)$.

(iv) The place weight function W is a **place flow** iff:
$(t,b) \in BE: \sum_{p \in P} W(p)(E(p,t)<b>) = \sum_{p \in P} W(p)(E(t,p)<b>)$.

We use $\mathbb{W}_P$ to denote the set of all place weight functions, while we use $\mathbb{W}_{PF} \subseteq \mathbb{W}_P$ and $\mathbb{W}_{PI} \subseteq \mathbb{W}_P$ to denote the set of those place weight functions that are place flows and which determine place invariants.

All weights are linear 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 mapping any weighted-set to the empty set. The weight function assigning 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 whereas a place flows is a static property, which 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.

Theorem 3.2
Let a CP-net be given. We then have:
(i) $\mathbb{W}_{PF} \subseteq \mathbb{W}_{PI}$.
(ii) No dead binding elements $\Rightarrow \mathbb{W}_{PF} = \mathbb{W}_{PI}$.

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. ♦

3.2.2 Extending Place Invariants to CP-nets with Channels

For CP-nets with channels we define place weight functions, weighted markings, place invariants, $\mathbb{W}_P$ and $\mathbb{W}_{PI}$ in exactly the same way as we did for CP-nets. Moreover, we define place/channel weight functions and place/channel flows as follows:

Definition 3.3
Let CCPN = (CPN, CS) be a CP-net with channels, where CPN = (Σ, P, T, A, N, C, G, E, I) and CS = (CH, CT, CE). We then define:
(i) W is a **place/channel weight** function, with range $R \subseteq \Sigma$:
$p \in P: W(p) \in [C(p)_{WS} \rightarrow R_{WS}]_L \quad ch \in CH: W(ch) \in [CT(ch)_{WS} \rightarrow R_{WS}]_L$.
(ii) For each marking M the place/channel weight function W determines $W(M) \in R_{WS}$:
$M \in \mathbb{M}: W(M) = \sum_{p \in P} W(p)(M(p))$.
(iii) The place/channel weight function W determines the **place invariant** $W(M) = W(M_0)$ iff the weighted marking is constant for all reachable markings:
$M \in [M_0\rangle: W(M) = W(M_0)$.

(iv) The place/channel weight function W is a **place/channel flow** iff:

(t,b) BE:

$$\sum_{p \in P} W(p)(E(p,t)<b>) + \sum_{ch \in CH} W(ch)(E(t,!?,ch)<b>) =$$

$$\sum_{p \in P} W(p)(E(t,p)<b>) + \sum_{ch \in CH} W(ch)(E(t,?!,ch)<b>).$$

We use $\mathbb{W}_{PC}$ to denote the set of all place/channel weight functions, while we use $\mathbb{W}_{PCF}$ $\mathbb{W}_{PC}$ and $\mathbb{W}_{PCI}$ $\mathbb{W}_{PC}$ to denote the set of those place/channel weight functions that are place/channel flows and which determine place invariants.

The domain of a weight function is extended to cover both places and channels, i.e., each place and each channel are mapped into a weight. A weight of a place is a linear function from weighted sets over the colour set of the place to weighted sets over R. A weight of a channel is a linear function from weighted sets over the channel type to weighted sets over R. A place/channel weight function determines a place/channel flow if the weighted sum of the tokens consumed together with the values of the !?-expressions are equal to the weighted sum of the tokens produced together with the values of the ?!-expressions. This is the only non-symmetric use of !? and ?!. We have chosen to group !?-expressions with the input arcs, and ?!-expressions with the output arcs. We could just as well do the opposite. The definition of place invariants is equivalent to the definition given for CP-nets.

For a weight function W $\mathbb{W}_{PC}$ we use W½P to denote its restriction to P and we use $\mathbb{W}_{PCI}$½P to denote {(W½P) ½W $\mathbb{W}_{PCI}$}. This means $\mathbb{W}_{PCI}$½P = $\mathbb{W}_{PI}$.

In the following we show how the concepts of place invariant and place/channel flow of CP-nets with channels match the corresponding concepts for CP-nets. A star is used to indicate that a symbol refers to the equivalent CP-net. As an example we use W* $\mathbb{W}_P^*$ to denote a place weight function of CPN*.

Theorem 3.4

Let CCPN be a CP-net with channels, having CPN* as the equivalent CP-net. We then have:

(i) $\mathbb{W}_{PCF}$½P $\mathbb{W}_{PF}^*$.

(ii) $\mathbb{W}_{PCF}$ $\mathbb{W}_{PCI}$.

Proof:

Property (i): This property is shown by reformulating the place/channel flow of Def. 3.3 (iv) in terms of the behaviourally equivalent CP-net using the transformation specified in Def. 2.4. The proof can be found in [CD92].

Property (ii): We have that $\mathbb{W}_{PCF}$½P $\mathbb{W}_{PF}^*$ $\mathbb{W}_{PI}^*$. The first inclusion follows from Th. 3.4 (i) and the second from Th. 3.2 (i). We conclude the proof of the property by noting that any extension of $\mathbb{W}_{PI}^*$ to cover channels is a place/channel invariant. ♦

As we will illustrate by the examples in Sec. 4 it is possible to determine the interesting place invariants of a CP-net with channels from the place/channel flows, it is however the case that certain use of channel communication can make it impossible to find place/channel invariants corresponding to all place invariants.

In the example of Fig. 3.1 and 3.2 we can specify three weight functions which determines place invariants:

W_1: $P_1(A + B)$
W_2: $P_2(A + B)$
W_3: $Id(A + B)$.

W_1 and W_2 specifies that the multi-set sum of the projection of the two places is constant for all reachable markings. W_3 specifies that the multi-set sum of the markings of the two places are constant. It can be verified from fig. 3.2 that W_1, W_2 and W_3 are place flows of the behaviourally equivalent CP-net.

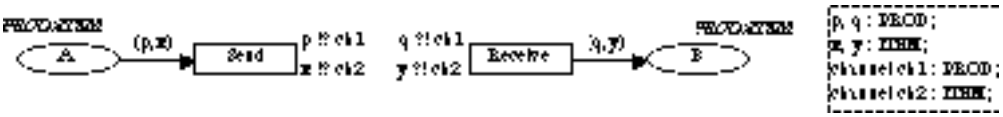

Figure 3.1: CP-net with channel communication

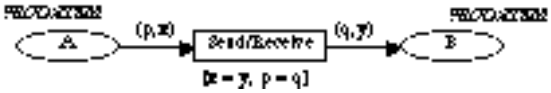

Figure 3.2: Behaviourally equivalent CP-net

It is possible to find place/channel flows which corresponds to W_1 and W_2:

W_1: $P_1(A + B) + Id(ch1)$
W_2: $P_2(A + B) + Id(ch2)$.

But it is not possible to find any weights of ch1 and ch2 corresponding to W_3.

4. Examples of CP-nets with Channels

In this section we illustrate the modelling convenience of CP-nets with channels by means of small examples. We focus on how channels can be used to glue together different sub-models into larger models.

The straightforward use of CP-nets with channels is to describe the synchronous communication between separate processes. Channels are then used to define the interface through which the processes communicate by means of communication transitions. This approach has a number of advantages. The most obvious advantage is that it is possible to structure the model into communicating sub-models in such a way, that you can change parts of the model without changing other parts of the model as long as the interface specified by means of channels is not changed.

4.1 A Resource Sharing Example

In the following we show how the resource allocation system in Fig. 4.1, [Jen92], can be re-structured using channel communication. The resource allocation example has a set of processes that 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, called 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 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 = Integer, U = P×I and R = {r,s,t}. We use a variable x of type P and a variable i of type I. 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, 3 q-processes, 1 r-resource, 3 s-resources and 2 t-resources. The CP-net is presented in Fig. 4.1 is so small, that we would not decompose it in practice, but it can still be used to illustrate the convenience of CP-nets with channels.

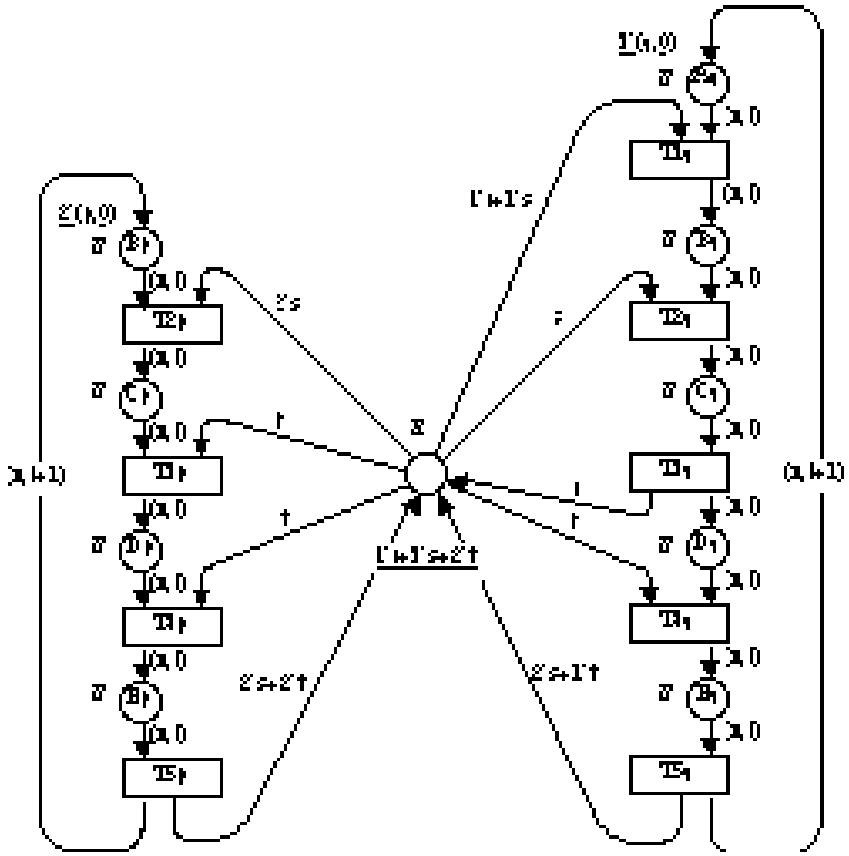

Figure 4.1: Resource allocation modelled with CP-nets

The idea is to re-structure the CP-net model of Fig. 4.1 into three separate sub-models communicating through channels—sub-models describing the p-processes, the q-processes and the resource allocation respectively. We use channels instead of ordinary arcs to reserve and release resources. Hereby the model is broken up into three separate submodels with interface described by means of channels. The sub-models are shown in Fig. 4.2.

The resource allocation sub-model consists of the resource place and two communication transitions GetResource and PutResource communicating through channels called Reserve and Release. The channel type of Reserve and Release is multi-sets over the colour set R and res is a multi-set variable over the colour set R.

In the subnets modelling the processes we replace the transitions normally connected to the resource place with communication transitions. Transitions reserving a resource are replaced with communication transitions using the Reserve channel and transitions releasing a resource are replaced with communication transitions using the Release channel.

Note that we allow a multi-set to be communicated through a channel, but we do not allow a multi set from one sender to be split up into pieces received by different receivers.

Given the model of Fig. 4.2 consisting of three separate sub-models, it is now possible to, e.g., redefine the resource allocation strategy without changing the process descriptions.

On the other hand the figure also illustrates that breaking a small example into sub-models by replacing some of the arcs with channels does not always enhance the readability of the model. The real advantage of CP-nets with channels will be gained when modelling large systems where it is impossible to model the whole system on a single page.

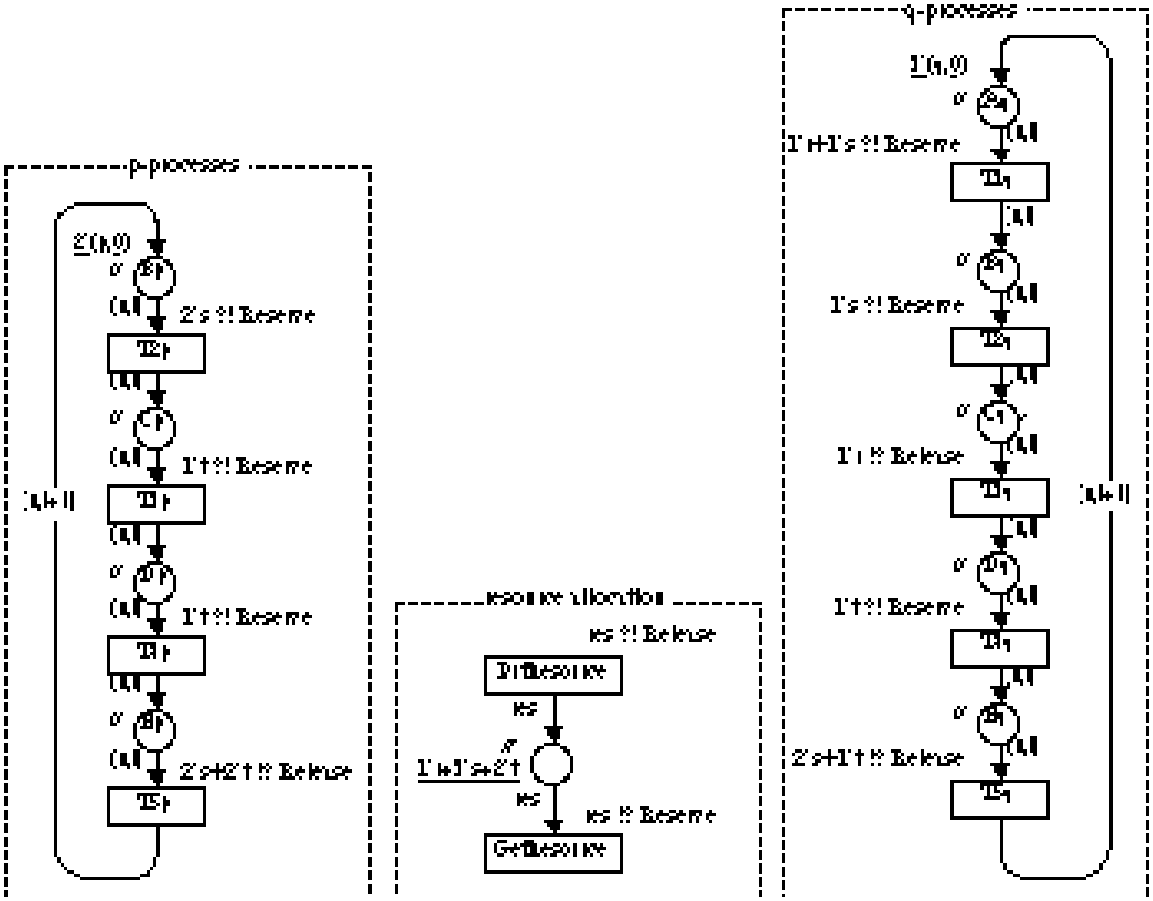

Figure 4.2: Resource allocation modelled with channels

Before we look at the place/channel flows of the CP-net with channels shown in Fig. 4.2 we look at the place flows of the CP-net in Fig. 4.1: We use the following notation for weight functions: 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×I into multi-sets of P, the Ig (Ignore colour) function maps a multi-set of size s into s`e, the indicator functions $I_r$, $I_s$, $I_t$ map multi-sets of resources into the size of the respective resources, i.e., $I_r$(2`r + 1`s + 3`t) = 2`e, $I_s$(2`r + 1`s + 3`t) = 1`e and $I_t$(2`r + 1`s + 3`t) = 3`e. Finally, we use the names 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. 4.1 we then have the following five place weight functions. By checking that the weighted sum of tokens consumed by each binding for each transition is equal to the weighted sum of tokens produced it can easily be shown that the weight functions are place flows and therefore determine place invariants.

W_1: Pr(Bp + Cp + Dp + Ep)
W_2: Pr(Aq + Bq + Cq + Dq + Eq)
W_3: I_r(R) + Ig(Bq + Cq)
W_4: I_s(R) + Ig(Bq) + 2 Ig(Cp + Dp + Ep + Cq + Dq + Eq)
W_5: I_t(R) + Ig(Dp + Eq) + 2 Ig(Ep).

We can construct other place flows, but all of the above place flows can easily be interpreted in terms 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 flows above, it is straightforward to prove that the system is deadlock free and similar behavioural properties.

Lets now consider the place/channel flows of the CP-nets with channels in Fig. 4.2. A place/channel weight function is a place/channel flow if the weighted sum of the tokens consumed together with the values of the !?-expressions are equal to the weighted set of tokens produced together with the values of the ?!-expressions. Therefore the weight functions, W_1 and W_2, which do not involve the resources are unchanged, whereas the other weight functions, W_3, W_4 and W_5, are extended to include the Reserve and Release channels.

W_1: Pr(Bp + Cp + Dp + Ep)
W_2: Pr(Aq + Bq + Cq + Dq + Eq)
W_3: I_r(R + Reserve + Release) + Ig(Bq + Cq)
W_4: I_s(R + Reserve + Release) + Ig(Bq) + 2 Ig(Cp + Dp + Ep + Cq + Dq + Eq)
W_5: I_t(R + Reserve + Release) + Ig(Dp + Eq) + 2 Ig(Ep).

4.2 Protocol Modelling

Although ordinary Petri nets has shown to be very useful in the area of protocol specification, CP-nets with channels may be used in this area with advantage. Often protocols are divided into a number of layers like, e.g., the OSI model, where each layer communicates only with the layer above and below. Using CP-nets with channels each layer is then modelled separately and the interfaces between the layers are modelled by means of channels. Hereby it is possible to model each layer independently and possibly at different abstraction levels. At the lowest level of details a layer can be modelled by means of two communication transition as shown in Fig. 4.3, where each communication transition handles the communication in one direction. At this abstraction level, the layer is transparent—messages are sent unchanged to the layers above and below. If it is later on decided to detail the description of this protocol layer, the communication transitions are just replaced with a subnet describing the internal behaviour of the layer at an appropriate level of abstraction.

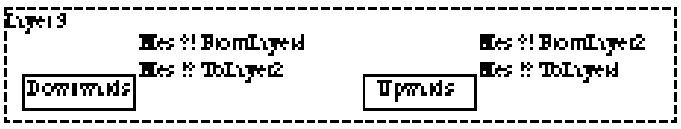

Figure 4.3: Transparent protocol layer

Using channels to describe the interface between two layers furthermore eases the description of, e.g., an unreliable communication. If we want to describe that data may be lost on the channel 'ToLayer2' in Fig 4.3 we do not need to change the original model, but just add a single transition occasionally removing data transmitted via the channel, as illustrated in Fig. 4.4.

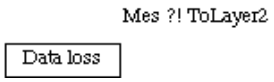

Figure 4.4: Modelling data loss on a channel

In this section we have illustrated how CP-nets with channels provide useful concepts for generating structured net models. We have illustrated how channels are useful to describe synchronous communication between separate sub-models and to glue separate sub-models together into larger models with extended functionality.

5. Conclusion

In this paper we have shown how the concept of synchronous communication channels can be introduced in the framework of CP-nets. CP-nets with channels facilitates the creation of compact and comprehensive models of systems with synchronous communication.

Channel communication is a valuable concept for structuring net models. Defining the interface between different subnets by means of communication channels makes it easy to create expandable and tailorable sub-models, where all dependencies between sub-models are expressed in the communication expressions. The advantage of such a modular approach has been known in the area of traditional programming languages for a long time, and has resulted in a number of different concepts, e.g., abstract data types and modules. CP-nets with channels have been developed to support similar efforts inside the area of Petri nets.

CP-nets with channels are highly influenced by needs originating from practical use of hierarchical CP-nets for modelling, and we have shown how CP-nets with channels fit nicely into the Petri net framework. We have shown that each CP-net with channels can be mapped into an equivalent CP-net, i.e., a CP-net with identical behaviour. We have defined place flows and place invariants for CP-nets with channels, in such a way that they have similar properties as in CP-nets.

Acknowledgement

We would like to thank Kurt Jensen for valuable discussions and Peter Huber, Mogens Nielsen and Laura Petrucci for many helpful comments on earlier versions of this paper.

References

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

[CP92] S. Christensen, L. Petrucci: **Towards a modular analysis of coloured Petri nets**. In: K. Jensen (ed.): Application and Theory of Petri Nets 1992. Lecture Notes in Computer Science, vol. 616, Springer-Verlag, 1992, 113-133.

[HT91] T. Hildebrand and N. Trèves: **S-CORT: A method for the development of electronic payment systems.** In: G. Rozenberg (ed.): Advances in Petri Nets 1989, Lecture Notes in Computer Science vol. 424, Springer-Verlag 1990, 262-280.

[Hoa85] C. A. R. Hoare: **Communicating sequential processes** ISBN 0-13-153289-8, Prentice Hall, 1985.

[HJS90] P. Huber, 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, 342-416. Also in [JR91], 215-243.

[Jen81] K. Jensen: **Coloured Petri nets and the invariant method.** Theoretical Computer Science 14 (1981), Springer-Verlag 1981, 317-336.

[Jen86] K. Jensen: **Coloured Petri nets.** In: W. Brauer, W. Reisig and G. Rozenberg (eds.): Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986 Part I, Lecture Notes in Computer Science, vol. 254, Springer-Verlag 1987, 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, 342-416. Also in [JR91], 44-119.

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

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

[Mil89] R. Milner: **Communication and concurrency**. ISBN 0-13-114984-9, Prentice Hall, 1989.