SICS Research Report
R86001
ISSN 0283-3638

Behavior-preserving Reductions of Communicating System Nets

Michael Yoeli and Björn Pehrson

Swedish Institute of Computer Science
Box 1263, SE-164 29 Kista, SWEDEN

1986

Among the various methods for the modeling of distributed systems that currently are available, the process algebra CCS and Petri nets with extensions are of particular interest. CCS contains the useful concept of synchronized communication. While the theory of CCS is based on a single concept of behavioral equivalence, a variety of such concepts could be incorporated into net theory. Particularly, a clear distinction can be made between the interleaving of concurrent actions and true concurrency. Net theory also provides a suitable basis for the investigation of structural properties (eg. liveness). Furthermore, various reduction techniques applicable to nets compare favorably with the algebraic analysis methods of CCS. In view of these arguments, a CCS-oriented extension of the Petri net concept, called Communication System Nets, CS-nets, is introduced which incorporates a conditional action feature. Various behavior-preserving reduction techniques are demonstrated along with the applicability of CS-Nets to the specification and verification of distributed systems.

BEHAVIOR-PRESERVING REDUCTIONS OF COMMUNICATING SYSTEM NETS

by

Michael Yoeli and Björn Pehrson

INTRODUCTION

Various methods for the modeling of distributed systems are presently available. Of particular interest are CCS [Mil80], [Mil83] and Petri nets [Bra80], [Pet81] as well as suitably extended net concepts [Kel76], [Yoe82]. Essential in CCS is the abstract concept of "synchronized communication" between processes, a useful concept which cannot be modelled directly by Petri nets (howewer, see [Gol-Myc84]). On the other hand, the theory of CCS is also based on a single concept of behavioral equivalence, whereas a variety of behavioral equivalence notions can be incorporated into net theory [Yoe-Etz 83]. Particulary, a clear distinction can be made between the interleaving of concurrent actions and true concurrency. Net theory also provides a suitable basis for the investigation of structural properties (e.g. liveness). Furthermore, various reduction techniques applicable to nets [Ber-Rou-Va79], [Peh83] compare favorably with the algebraic analysis methods of CCS.

In view of the above arguments a CCS-oriented extension of the Petri net concept, called "Communication-System (CS) Net" was introduced in [Peh-Yoe84] . It also incorporated the "conditional action" feature found in [Kel76], [Yoe82] and [Dia-Sil83]. We discussed various behavior-preserving reduction techniques and demonstrated the applicability of CS-nets to the specification and verification of distributed systems.

This paper is a further contribution to the theory of CS-nets and its applications. It uses a simplified version of the notion of CS-net. In order to make this paper self-contained, we repeat some of the material from [Peh-Yoe84], mostly in a somewhat modified way.

We include various additional behavior-preserving simplification methods and illustrate their applicability to the verification of a simple communication protocol.

We also show how the abstract CCS-concept of synchronized communication between processes can be implemented by means of self-timed signaling techniques [Mea-Con80].

In the concluding section of this paper we make some recommendations as to further research concerning CS-nets.

2. COMMUNICATING SYSTEM NETS

In this section we introduce the concept of CS-Net (Communicating Systems Net), which is a simplified version of the corresponding concept defined in [Peh-Yoe84]. We start by establishing our terminology and notations concerning Petri nets [Bra80], [Pet81].

DEFINITION 2.1: *A Petri Net* is a triple N = (P,T,V), where
(1) P and T are finite, disjoint sets of places and transitions, respectively
(2) $V:(P\times T)\cup(T\times P) \rightarrow \{0,1\}$
A *marked Petri net* is a pair (N,M) where N is a Petri net and M is a *marking* of N, i.e. a function $M:P \rightarrow \omega$, where ω denotes the set of non-negative integers.

In the graphical representation of Petri nets, places are represented by circles, transitions by bars, and V(x,y) = 1 corresponds to a directed arc from x to y. M(p) = k is indicated by k tokens inside the circle representing p.

For any transition $t \in T$, we set
$M\ [t\rangle :\Leftrightarrow (\forall p\in P)\ M(p) \geq V(p,t)$ and
$M\ [t\rangle\ M' :\Leftrightarrow M\ [t\rangle \wedge (\forall p\in P)\ M'(p) = M(p) - V(p,t) + V(t,p)$
The above notation is extended to finite sequences of transitions in the usual way.
The marking M' is *reachable* from M iff $M[w\rangle M'$ for some finite sequence $w\in T^*$

We shall also use the "dot-notation" for any $x\in P\cup T$:
$x^{\cdot} := \{y \mid V(x,y) = 1\}$
$^{\cdot}x := \{y \mid V(y,x) = 1\}$

A marked Petri net (N,M) is a *single-token net* (state machine) iff $\Sigma M'(p)=1$ for every marking M' reachable from M. It is k-bounded, $k \geq 1$, iff $M(p) \leq k$ for every $p\in P$ and every marking M' reachable from M. It is safe, iff it is 1-bounded.

DEFINITION 2.2: A *CS-Net* Γ consists of the following:
(1) A Petri Net N = (P,T,V); The set T is partitioned into $T = T_D\cup T_K$, $T_D\cap T_K=\varnothing$ where T_D is the set of *data transitions* and T_K is the set of *channel transitions*.
(2) A *data set* D, together with a *condition function* $C:D\times T_D \rightarrow \{\text{true,false}\}$ and an *action function* $A:D\times T_D \rightarrow D$
(3) A *channel set* K, together with a *channel function* $\eta:T_K\rightarrow K!\cup K?$, where $K! = \{k! \mid k\in K\}$ and $K? = \{k? \mid k\in K\}$.
A CS-system consists of a CS-Net Γ together with an *initial state* $q_0=(M_0,d_0)$, where M_0 is a marking of N and $d_0\in D$.

The graphical representation of a CS-net is obtained from the graphical representation of its underlying marked Petri Net (N,M_0) by labeling its transitions as described below. The labels are written in the bars representing the transitions to which they are associated.

A data transition $t \in T_D$ is labeled by $[c_t/a_t]$, where c_t is the unary predicate on D defined by $c_t(d) = C(d,t)$ for every $d \in D$ and $a_t: D \rightarrow D$ is defined by $a_t(d) = A(d,t)$ for every $d \in D$.
A channel transition $t \in T_K$ is labeled by $\eta(t)$. The initial data value $d_0 \in D$ is suitably specified.
The label $[true/a_t]$ may be replaced by $[/a_t]$. Similary, a_t may be omitted if a_t is the identity function I_D on D. Furthermore, the label τ will represent the label $[true/I_D]$.

DEFINITION 2.3: A *Condition-Action* (CA) Net is a CS-Net without channels, i.e. $K=\varnothing$ and $T_K = \varnothing$.

EXAMPLE: The following example of a CS-system (see Fig 2.1) represents a simple communication system. In this example, D is a Cartesian product, namely $D = X^\omega \times X \times X^*$ where X denotes a given set of (feasible) messages, X^ω is the set of all infinite sequences of messages in X, and X^* is the set of all finite sequences of X-messages, including the empty sequence λ.

Any data value $d \in D$ will be represented as $d = (S,B,R)$, each component of this vector ranging over the corresponding component set of D. The initial data value d_0 is specified by $d_0 = (S_0,B_0,\lambda)$.
Given $S \in X^\omega$, we denote its i-th component by S[i], thus S = S[1], S[2], S[3],...
Furthermore, we denote by $S\downarrow$ the sequence $S\downarrow$:= S[2], S[3], S[4],...
Finally, we use "·" to denote concatenation.

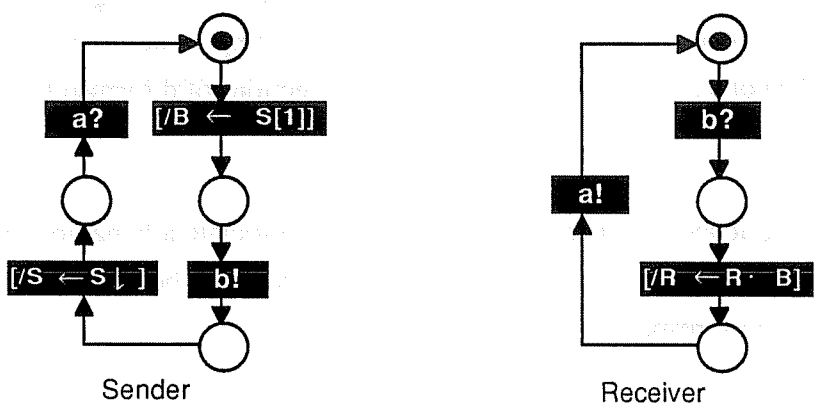

Figure 2.1 A CS-system representing a simple communication system

DEFINITION 2.4: A *state* of a CS-Net Γ is a pair q = (M,d), where M is a marking of N and $d \in D$.

A data transition $t \in T_D$ is fireable in state q = (M,d) if
(i) M [t> holds for N
(ii) C(d,t) = true
Let $t \in T_D$ be fireable. The outcome of firing t is the new state q' = (M',d') where M' is the marking obtained by firing t in (N,M), i.e. M [t> M', and d'= A(d,t).
Let $t,t' \in T_K$. If $\eta(t) = k!$ and $\eta(t') = k?$ for some $k \in K$ then t and t' are cotransitions. A channel transition may fire only in conjunction with one of its cotransitions. A pair {t,t'} of cotransitions is fireable in state q = (M,d) if
$(\forall p \in P)\ M(p) \geq V(p,t) + V(p,t')$
In this case the outcome of firing the pair {t,t'} is the new state q'= (M',d') where d'= d and
$(\forall p \in P)\ M'(p) = M(p) - V(p,t) - V(p,t') + V(t,p) + V(t',p)$.
The state q' is a successor of state q if q' can be obtained from q by firing either a data transition $t \in T_D$ or a pair of cotransitions $\{t,t'\} \in T_K$. We denote the successor relation by $\rightarrow$.

3. BEHAVIOR OF CS-NETS

Various notions of "behavior" can be defined for CS-systems, cf [Yoe-Etz83]. In this paper we follow [Peh-Yoe84] and restrict our considerations to notions of behavior particularly suitable for the description and verification of communication protocols.

DEFINITION 3.1: Let Γ be a CS-Net. A sequence of Γ-states $q_0,q_1,..,q_n$ is an *execution sequence* of the CS-system (Γ,q_0) if $q_i \rightarrow q_{i+1}$ for i = 0=1,...,,n-1.

Assume D is a Cartesian product, say $D = D_1 \times D_2 \times .. \times D_m$. Let y_i denote a variable ("data-variable") ranging over D_i (i=1,2,..,m). Let y be a subset of $\{y_1,..,y_m\}$ containing the y_i-variables which are in some specified sense "essential" (e.g. input and output variables).
For any state q = (M,d) of Γ, let y(q) be the subvector (projection) of d formed by its essential components.

With any execution sequence $w = q_0,q_1,...,q_n$ of (Γ,q_0) we associate a behavior sequence with respect to y, namely $w_y = y(q_0),y(q_1),...y(q_n)$. The set B_y of all such behavior sequences is called the behavior of (Γ,q_0) with respect to y.

Behavior sequences, as defined above, frequently include redundant repetitions, which should be omitted, if a concise description of the behavior of the system is required. We formalize these concepts in the following definitions.

DEFINITION 3.2: Let Σ be a finite alphabet, and $\sigma \in \Sigma$. We denote by σ^n the sequence $\sigma\sigma..\sigma$ (n times). Let now $w=\sigma_1^{n_1} \sigma_2^{n_2}... \sigma_k^{n_k}$, where $\sigma_i \neq \sigma_{i+1}$ for i = 1,2,..,k-1 and $n_i > 0$ for i=1,..,k

The *repetition-free reduction* of w is the sequence red(w) = $\sigma_1\sigma_2...\sigma_k$.

DEFINITION 3.3: Let (Γ,q_0) be a CS-system, and B_y its behavior with respect to y. We define the *repetition-free behavior* RB_y of (Γ,q_0) with respect to y by $RB_y = \{red(w) \mid w \in B_y\}$.

DEFINITION 3.4: Two CS-systems are said to be *behavior-equivalent* with respect to y if their repetition-free behaviors with respect to y are the same. They are *d-equivalent* if they have the same data set and are behavior-equivalent with respect to all their data variables.

A given CS-system which is safe (or bounded) can be converted into a d-equivalent CA-system, where the underlying marked Petri net is a single-token net (finite-state machine). This "FSM-conversion" is illustrated, for the CS-system of figure 2.1, in figure 3.1(a). In this figure, $\tau(a)$ indicates a τ-labeled transition, which corresponds to the pair of cotransitions a!,a? in the given system.

The notation A1||A2 is used here to indicate the interleaving of actions A1 and A2, provided the outcome is deterministic (independent of the order of execution).

NOTE: The notation A1||A2 may also be interpreted as including the concurrent execution of A1 and A2, whenever the distinction between interleaving and "true concurrency" is of interest. (cf. [Yoe-Etz83], [Gol-Myc84]).

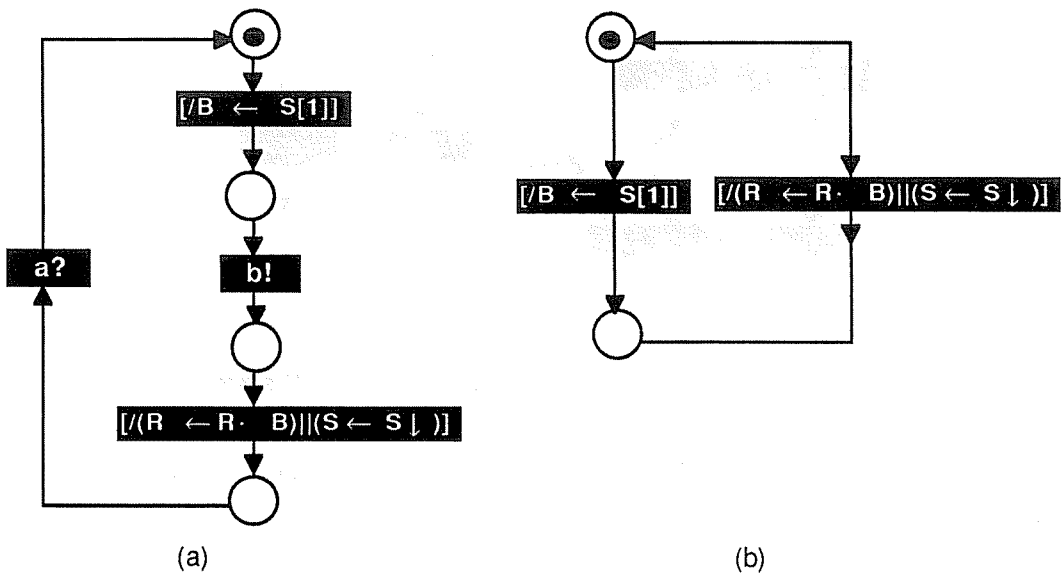

Figure 3.1 Conversion of the CS-net of figure 2.1 into a single-token net

Evidently, the CS-systems of figure 3.1 (a) and figure 3.1 (b) are both d-equivalent to the CS-system of figure 2.1.

The FSM-conversion of a bounded CS-system is a straight-forward application of the basic definitions. However, it may become computationally inefficient, if the given CS-system has a high degree of parallelism.

In the sequel it will be convenient to simplify the graphical representation of CS-systems, as shown in figure 3.2

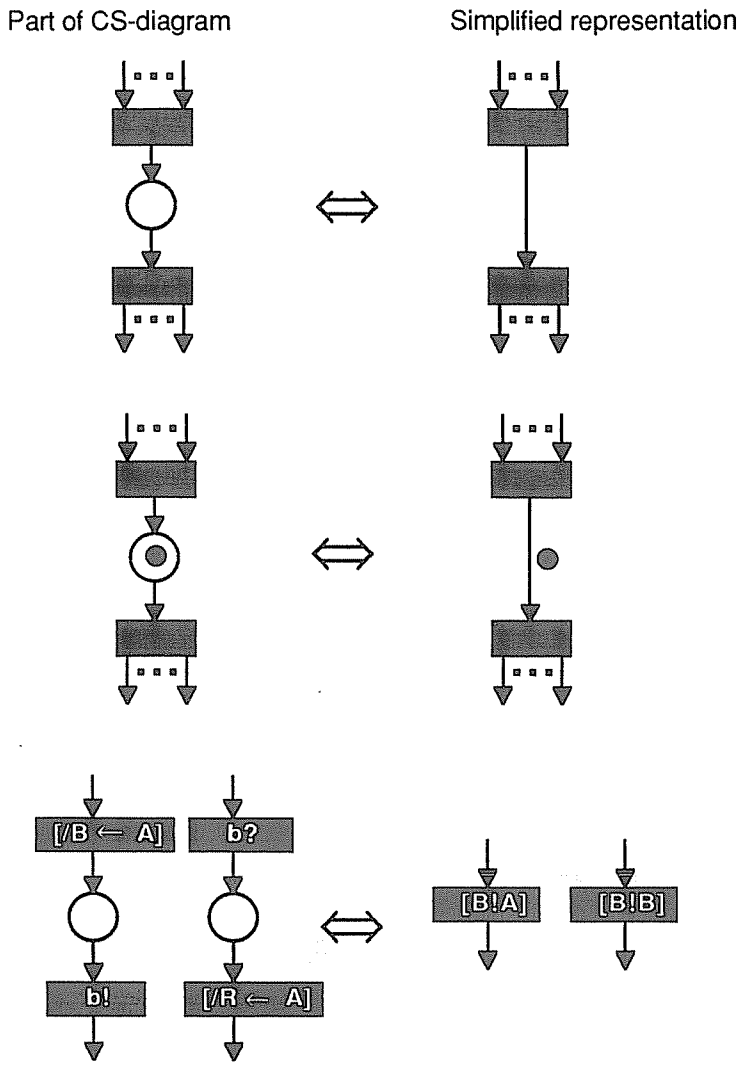

Figure 3.2 Simplified graphical representation of CS-systems

Using the notation of figure 3.2, we show in figure 3.3 three CS-systems which are all behavior equivalent with respect to (S,B,R) to the CS-system of figure 2.1.

Both figure 3.3 (b) and figure 3.3 (c) contain no channels, i.e. they represent CA-systems. Fig 3.3 (c) is derived from figure 2.1 by replacing its channel transitions by data transitions relating to non-essential (internal) binary ("signaling") data. This transformation indicates how the abstract concept of "synchronized communication" (which is basic in CCS [Mil80]) can be implemented by

a suitable "handshake" mechanism using binary flags. (See [Mea-Con80] for VLSI-oriented "self-timed signaling" methods).

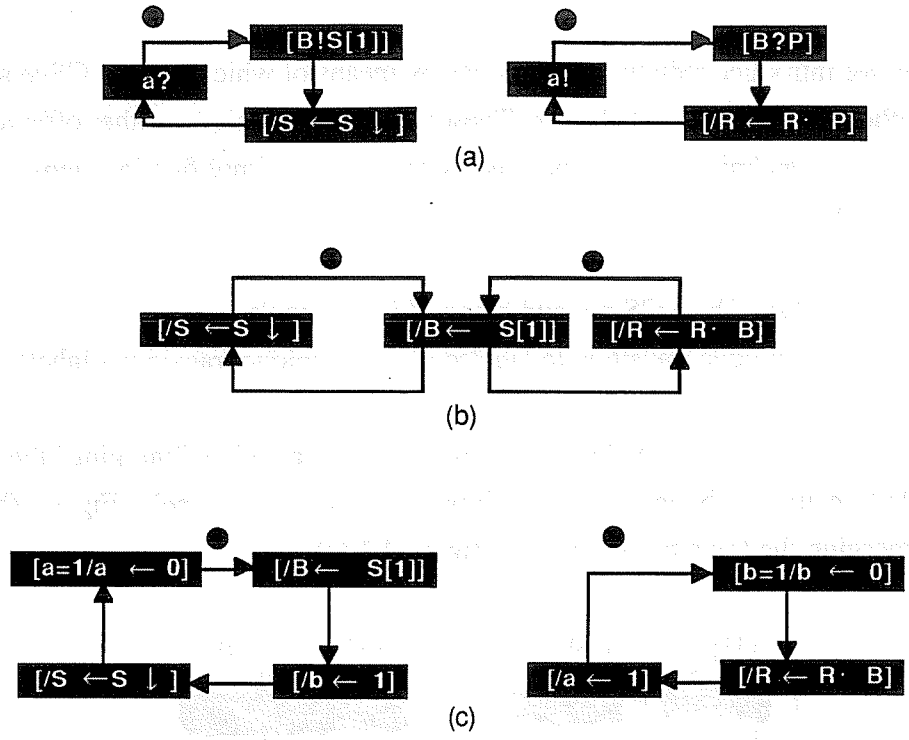

Figure 3.3

A more general tranformation rule is shown in figure 3.4. Provided the cotransitions labelled k! and k? in figure 3.4 (a) have no additional cotransitions (i.e. no transition inside A or B is labeled k! or k?), CS-systems of figure 3.4 (a) and figure 3.4 (b) will be bahavior-equivalent with respect to all the data variables of figure 3.4 (a).

We omit the formal proof of this statement, as well as a more general discussion of relevant transformation rules. See [Ro-Ro-Pnu84] for hardware implementations of CSP-Primitives and their verification.

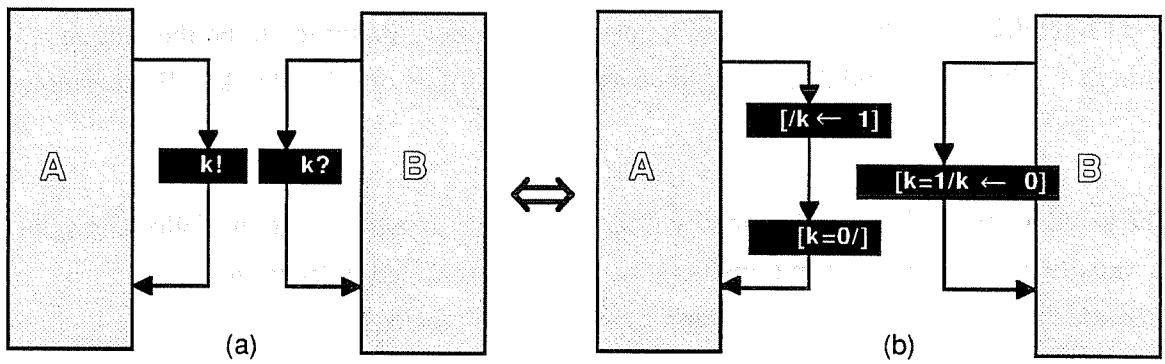

Figure 3.4

4. REDUCTION OF CS-SYSTEMS

In this section we introduce reduction techniques by means of which a given CS-system can be simplified without changing its d-behavior. These reduction techniques either offer an alternative to the FSM-conversion technique or may be used as preliminary simplification steps, prior to the application of FSM-conversion.

DEFINITION 4.1: Let Γ be a CS-net and k one of its channels.
Assume there exists a unique transition to labeled k! and a unique transition t labeled k? and that $\dot{t} \cap \dot{t}' = t\dot{} \cap t'\dot{} = \varnothing$.
Then the *k-reduction* of Γ, denoted by $\Gamma\backslash k$, is the CS-net obtained by "merging" the transitions t and t' into a single transition, labelled τ. This is shown in figure 4.1. Namely, Fig 4.1 (b) shows the outcome of merging the transitions t and t' of figure 4.1 (a).

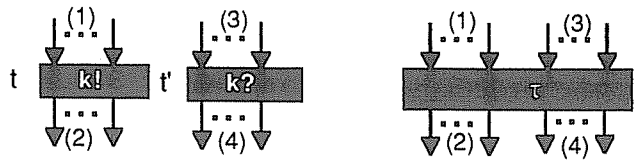

Figure 4.1

One easily verifies the following proposition.

PROPOSITION 4.1: Let (Γ, q_0) be a CS-system and $\Gamma\backslash k$ the k-reduction of Γ. Then the CS-systems (Γ, q_0) and $(\Gamma\backslash k, q_0)$ are d-equivalent.

We will now extend definition 4.1 to cases of *multiple cotransitions*.

DEFINITION 4.2: Let Γ be a CS-net and k one of its channels. Assume t to be the unique transition of Γ labeled k! , and $t'_1, t'_2, .., t'_m$ to be all the transitions of Γ labeled k?. Furthermore, let $\dot{t} \cap \dot{t}_i' = t\dot{} \cap t_i'\dot{} = \varnothing$ for i = 1,...,m.

Then the k-reduction of Γ (again denoted by $\Gamma\backslash k$) is obtained by performing the following steps.
(1) the transition t is "split" into m transitions $t_1,...,t_m$, as illustrated in figure 4.2.
(2) transition t_i (i = 1.,,,.m) is labeled k_i! and transition t_i' is labeled k_i?
(3) Let Γ' be the CS-net obtained by (1) and (2).
Then $\Gamma\backslash k = \Gamma'\backslash k_1 \backslash ... \backslash k_m$

Evidently, Proposition 4.1 is also valid for this extended definition of $\Gamma \backslash k$.

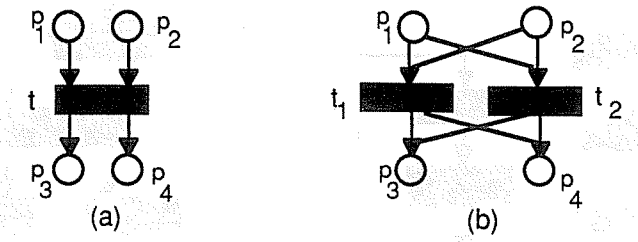

Figure 4.2 The net in (b) is obtained by "splitting" the transition t in (a)

The other cases of multiple cotransitions are treated similarly. We omit the formal details.
The elimination of channel transitions replaces them by transitions labeled τ. Such τ-transitions can be eliminated by means of suitable reduction rules. Examples of such rules are shown in figure 4.3 (a) and (b), where $[\Lambda_i]$ represents an arbitrary transition label. figure 4.3 (c) illustrates the elimination of a redundant place.

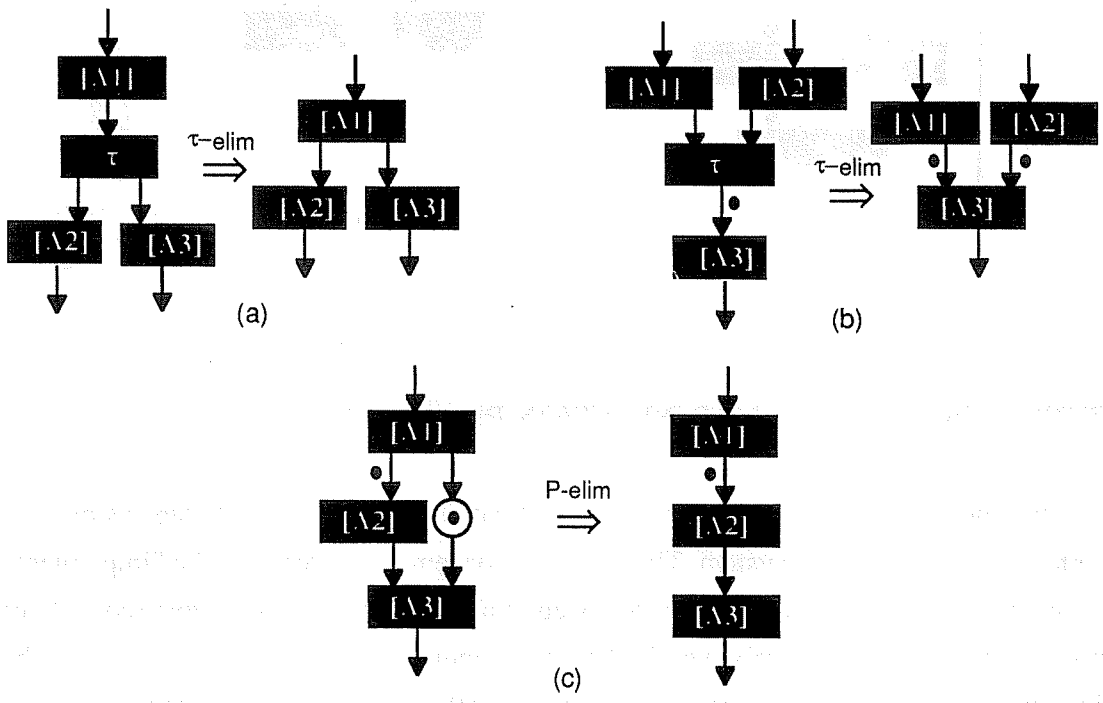

Figure 4.3

Figure 4.4 shows how the CS-system of figure 2.1 can be reduced to the d-equivalent system of figure 3.3 (b). Fig 4.4 (a) and figure 4.4 (d) evidently correspond to Fig 2.1 and figure 3.3 (b), respectively.

Figure 4.4 (b) is obtained from figure 4.4 (a) by a-reduction and b-reduction. The application of P-elimination (see figure 4.3 (c)) yields figure 4.4 (c). Finally, figure 4.4 (d) is obtained by eliminating both τ-transitions in accordance with the rules of figure 4.3 (a) and (b).

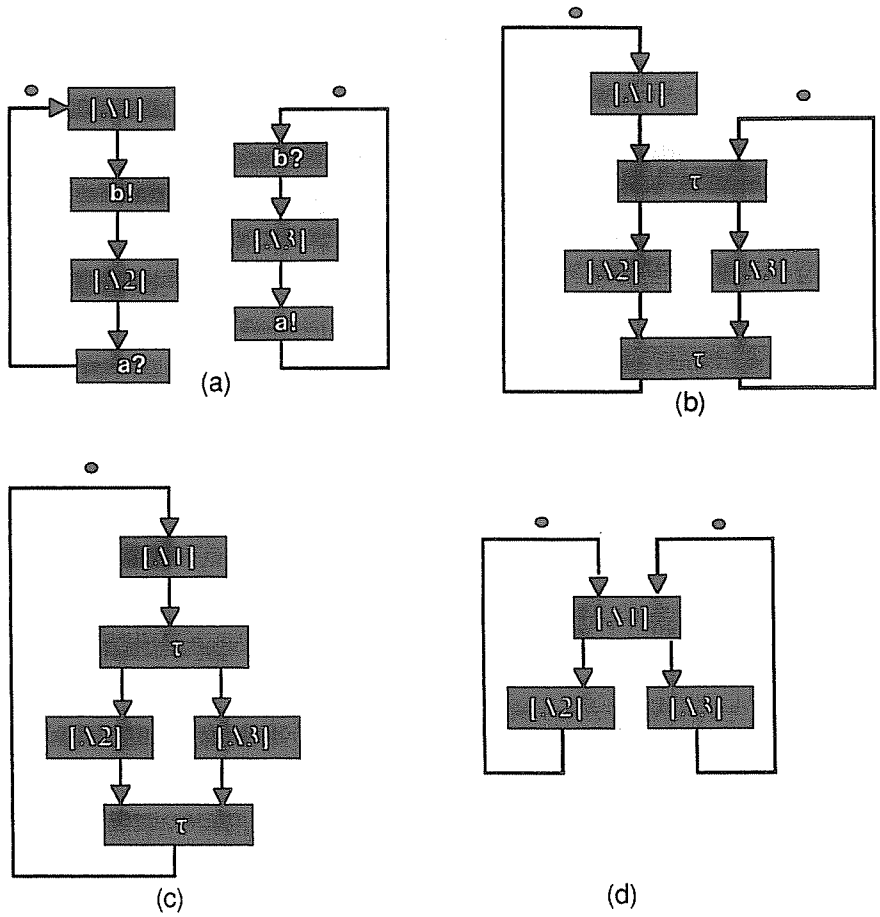

Figure 4.4

5. VERIFICATION OF A COMMUNICATION PROTOCOL

In this section we illustrate our approach to the verification of a given distributed system, represented by means of a CS-system. The given CS-system (considered as the "implementation") may be verified by proving it to be behaviorally equivalent to a given CA-system (considered as the system specification). Alternatively, verification may require proving that a given list of behavioral specifications is satisfied by the given CS-system. In both cases a behavior-preserving simplification of the given CS-system will frequently lead to its immediate verification.
Such a behavior-preserving simplification will in general consist of the application of reduction techniques, combined with FSM-conversion.

For the purpose of illustrating the above concepts, we consider a communication protocol represented by the CS-system of figure 5.1 (b). Its block diagram is shown in figure 5.1 (a). This system is similar to the one shown in figure 2.1, and the notation introduced in figure 2.1 is used here again.

However, we now assume an unreliable MEDIUM between SENDER and RECEIVER. The MEDIUM may introduce errors into the message it transfers from data link B to data link H. We assume a suitable error-detection code is used in the system. We denote by E an erroneous message, not in the given set X of (error-free) messages. The RECEIVER is capable of determining whether a received message is error-free (i.e. an element of X).

The only assumption we make about the system specification is that it is only concerned with the variables S and R. Therefore, we are interested in a simplification of the given system which preserves the (S,R)-behavior.

Figure 5.2 is obtained from figure 5.1 by the introduction of the label symbols $[\Lambda_i]$, as shown in the figure. This will facilitate the discussion of the various simplification steps.

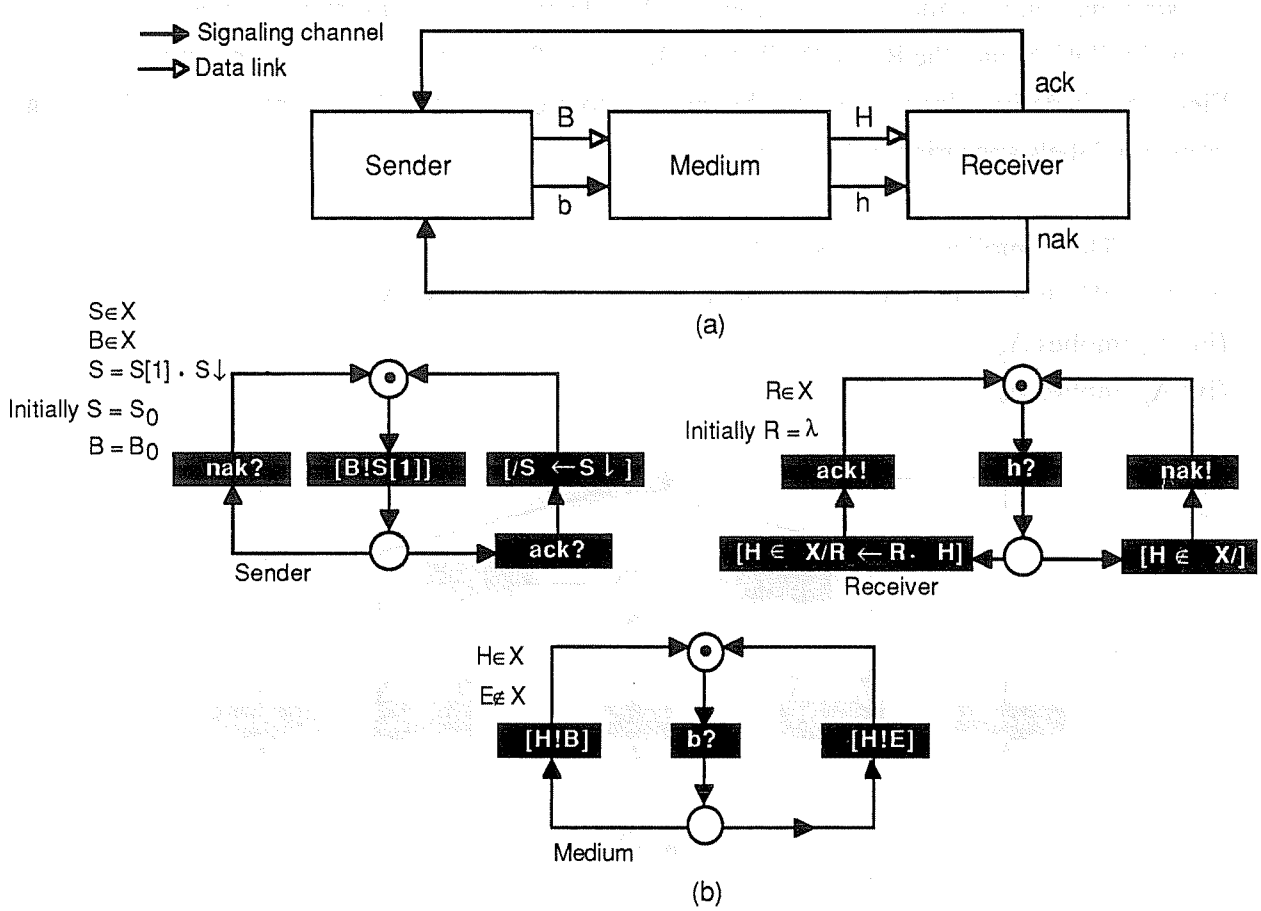

Figure 5.1 (a) 3-component communication protocol and (b) its representation as CS-system

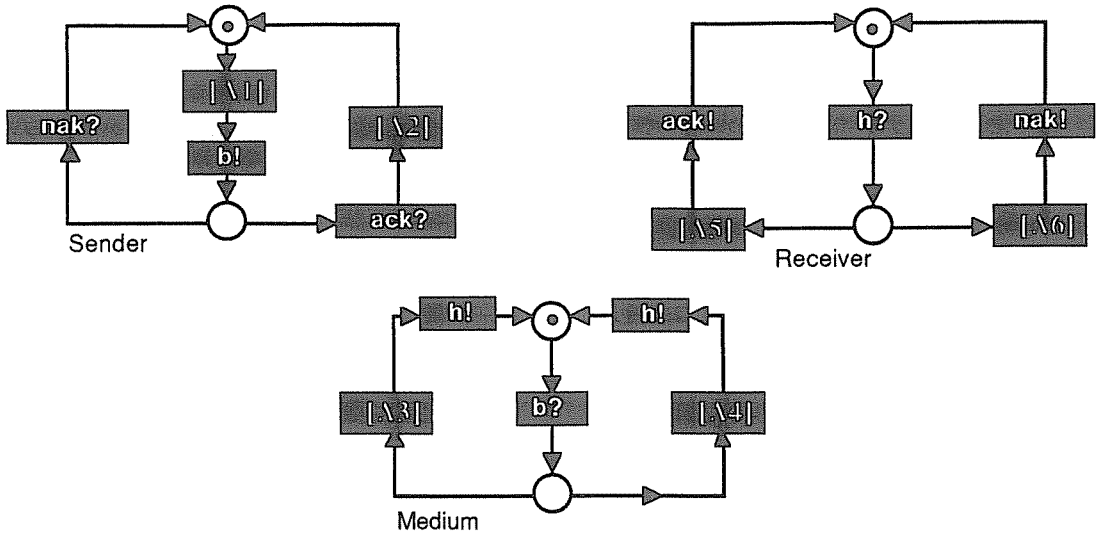

Figure 5.2

We start our simplification steps by applying h-reduction to the subsystem of figure 5.2 consisting of the MEDIUM and the RECEIVER (see Def. 4.2). The outcome is shown in figure 5.3. Figure 5.4 is derived from figure 5.3 by taking into account the following relationship between transition labels appearing in figure 5.3:

(i) $\Lambda_3 = /H \leftarrow B$ implies $\neg\Lambda_6 = \neg H \notin X$
(ii) $\Lambda_4 = /H \leftarrow E$ implies $\neg\Lambda_5(C) = \neg H \in X$, where $\Lambda_5 = \Lambda_5(C)/\Lambda_5(A)$
(iii) Λ_3 implies $\Lambda_5(C)$
(iv) Λ_4 implies Λ_6.

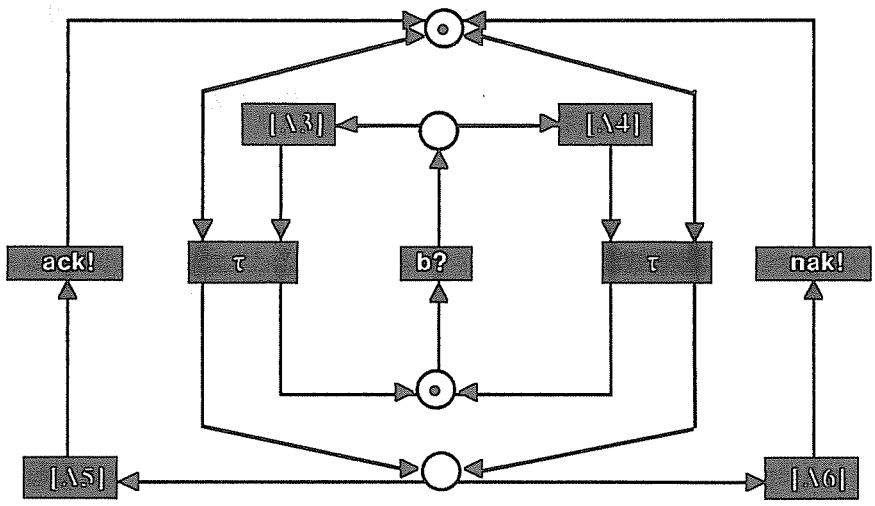

Figure 5.3

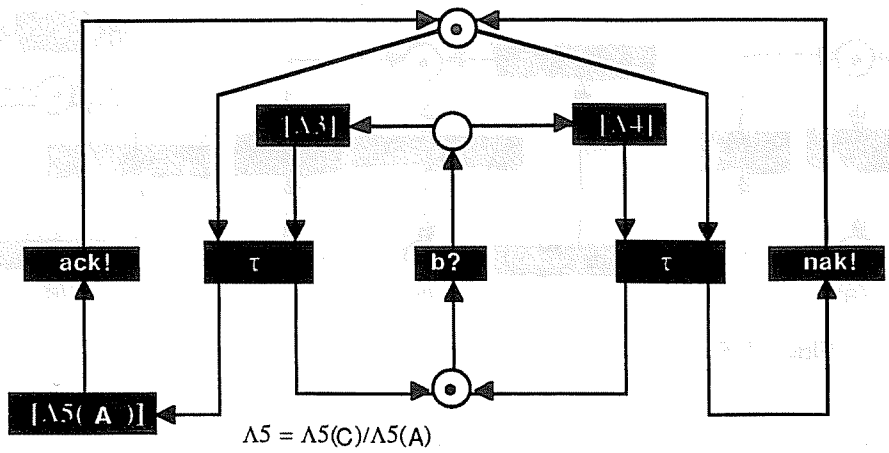

Figure 5.4

The transformation of figure 5.4 into figure 5.5 illustrates an instance of "place-splitting". One easily verifies that the d-behavior of the overall system is not changed by this transformation.

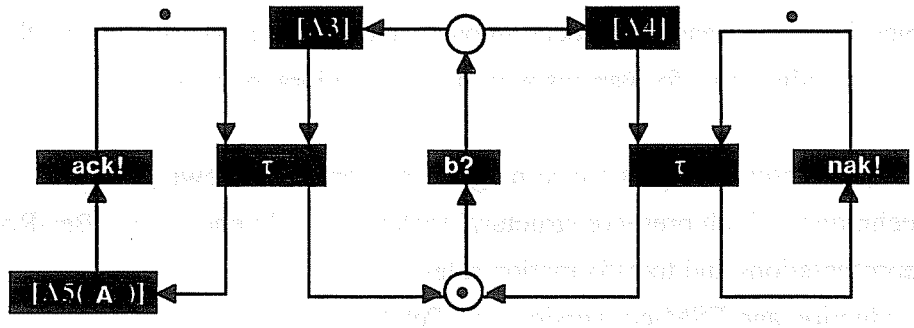

Figure 5.5

It is now very easy to apply FSM-conversion to the S-system consisting of the SENDER-component of figure 5.2 and the reduced MEDIUM-RECEIVER subsystem of figure 5.5. The outcome of this FSM-conversion is shown in figure 5.6 (a). Eliminating non-essential variables, we arrive at the single-token CA-systems of figure 5.6 (b) and figure 5.6 (c). Here transitions labeled τ represent actions which do not involve changes of the essential variables.

These τ-transitions may only be removed if the MEDIUM-component is assumed to be "fair", i.e. will never become completely unreliable. However, a more formal discussion of this topic is beyond the scope of this paper (cf. [Tom-Yoe81], [Par-Gus84]).

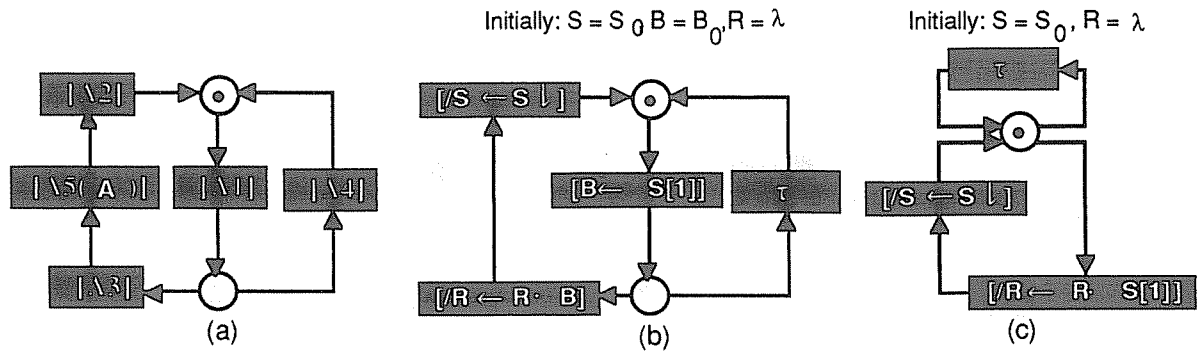

Figure 5.6

CONCLUSIONS

Further to [Peh-Yoe 84], we have demonstrated in this paper the applicability of the CS-net model to the description and behavioral verification of distributed systems, particularly communication protocols.

However, we have only indicated various behavior-preserving reduction techniques by means of suitable examples. Further research is needed in order to establish a comprehensive theory of behavior-preserving reductions, for various notions of behavioral equivalence.

Further recommended research topics concerning CS-nets are the following:
(a) reduction techniques which preserve structural features (e.g. liveness): cf. [Ber-Rou-Va79].
(b) algebraic representations and transformation rules
(c) automated reduction and FSM-conversion, cf. [Peh 83].
(d) system descriptions using CS-nets in combination with temporal logic: cf. [Tom-Yoe81], [Dia-Sil83], [Par-Gus84].

7. REFERENCES

[Ber-Rou-Va79] G. Berthelot, G Roucairol, R. Valk: Reduction of Nets and Parallel Programs, in[Bra80]:

[Bra80] W. Brauer (ed): Net Theory and Applications, LNCS 84, 1980.

[Dia-Sil83] M. Diaz, G. Guidacci Da Silveira: Specification and Validation of Protocols by Temporal Logic and Nets, Proc. IFIP Congress, Paris 1983.

[Gol-Myc84] U. Goltz, A. Mycroft: On the Relationship of CCS and Petri Nets, LNCS 172: Proc. ICALP 1984.

[Kel76] R.M. Keller: Formal Verification of Parallel Programs, CACM 1976, pp. 371-384.

[Mea-Con80] C. Mead, L. Conway: Introduction to VLSI Systems, Addison-Wesley, 1980.

[Mil80] R. Milner: A Calculus of Communicating Systems, LNCS 92, 1980.

[Mil83] R Milner: Calculi for Synchrony and Asynchrony, TCS, Vol 25, 1983, pp. 267-311.

[Par-Gus84] J. Parrow, R. Gustavsson: Modeling Distributed Systems in an Extension of CCS with Inf. Experiments and Temporal Logic, Proc. 4th Int. Conf. on Protocol Spec., Verif. and Testing, Skytop 1984.

[Peh83] B. Pehrson: Abstraction by Structural Reduction, Proc. 3rd Int. Conf. on Protocol Spec., Verif. and Testing, Zurich 1983.

[Peh-Yoe84] B. Pehrson, M. Yoeli: A Communication System Net Model for Spec. and Verif. of Distr. Systems, Proc. 4th Int. Conf. on Protocol Spec., Verif. and Testing, Skytop 1984.

[Pet81] J. L. Peterson, Petri Net Theory and the Modeling of Systems, Prentice-Hall, 1981.

[Ro-Ro-Pnu84] D. Ron, F. Rosenberg, A Pnueli: A Hardware Implementation of the CSP Primitives and its Verification, LNCS 172: Proc. ICALP 1984.

[Tom-Yoe81] A. Tomer, M. Yoeli: On the Application of Extended Petri Nets to the Verif. of Protocols, TR 248, Computer Science Dept., Technion, Haifa, 1981.

[Yoe82] M. Yoeli: Synthesis of Concurrent Systems, IF 52: Selected papers from 1st and 2nd Europ. Workshop on Petri Nets. 1982.

[Yoe-Etz83] M. Yoeli, T. Etzion: Behavioral Equivalence of Concurrent Systems, IF 66: Selected papers from 3rd European Workshop on Petri Nets, 1983.

IF = Informatik Fachberichte, Springer Verlag
LNCS = Lecture Notes in Computer Science, Springer Verlag