International Journal of Theoretical Physics, Vol. 21, No. 12, 1982

Modular Implementation of Concurrency

Lutz Priese

Fachbereich Mathematik-Informatik, Universität, D-4790 Paderborn, Federal Republic of Germany

Received May 8, 1981

We try to develop a deeper understanding of "concurrent computations" by stating a precise and formalized conception of a realization of a concurrent computation. Within this formal apparatus we can present exact mathematical proofs for certain hierarchies of concurrent systems. As the following systems have a modular structure they might be helpful for a further theory of very fast hardware circuits without a global synchronizing clock.

1. INTRODUCTION

In recent developments concurrency becomes a more and more important feature of computers. On the software side first attempts have been made to design concurrent languages and automatic translations from sequential into parallel programs. On the hardware side a global concurrent behavior is quite common. However, on the microscopic level of hardware we still follow the ancient approaches of "synchronized" Boolean logic. For a future design of very fast computers we at once meet some physical constraints: the faster we operate the more uncertain becomes location and propagation of signals, stability and switching time of gates, etc., with the unwanted and contradictory consequence of delaying a global clock for these processes for security reasons.

A quite radical escape of this circle is a consequent renunciation of synchronization. Instead of a global time we may imagine the time ordering as a lattice. For switching circuits this implies that any wire and gate has its own local time. A logical design for such circuits should ensure that the overall behavior does not become destroyed by local time properties, such as propagation speed of signals or switching time of gates—with the consequence that we do not have to delay any operation for security reasons!

0020-7748/82/1200-0993$03.00/0

Also one should not operate with hand-shaking signals that cost too much waiting time.

Such a consequence is not as severe as it may appear at a first glimpse, owing to our quite good understanding of "asynchronous" logics from the work of many researchers, where especially C. A. Petri has to be named. Petri nets are well developed and understood, and have become the most important model of an asynchronous logic today.

However, an approach to implement Petri nets into a hardware design meets some difficulties. Firstly: it is by no means clear what a concurrent computation or a realization of such a computation is. Secondly: Petri nets have no modular structure, as a solving of conflicts is not explicitly expressed in the nets and may involve unbounded parts of the whole structure. The aim of this paper is a contribution to a theory of modular concurrency. This approach and motivation is not new. There are lots of papers on asynchronous modules—let us just mention the previous works of Dennis, Patil, and Furtek at MIT. However, all these previous papers have a certain lack of mathematical formalism and preciseness. This statement should become more evident if one regards, for example, the important impossibility results of Patil (1971) ("Not any Petri net can be realized by a simple net") and Kosaraju (1973) ("Not any inhibitory Petri net can be realized by a Petri net"), that present impossibility proofs *without* a precise definition of the concept of a realization. Also one has to be very careful with conceptions of computation. There are the well-known concepts of "weak" and "strong" computability of Petri nets by Hack (1976) and Lipton (1976), that are very powerful for certain proofs on reachability properties, but that give no insight into computation properties of asynchronous systems at all. The same holds for language conceptions.

Fortunately there are some recent attempts to clarify the conception of a realization (or simulation) of a concurrent computation by Jensen (1980), Kasai and Miller (1979), Kwong (1977), and Berthelot, Roucairol, and Valk (1979). But these papers present very general properties of realizations and do not consider the peculiarities of concurrency and thus might not help in an understanding of concurrent computations.

The plan of this paper is to present a (hopefully) reasonable concept of realization of concurrent systems (Section 2) and to present several nontrivial results on modular realizations of concurrent systems (Section 3) that should help to develop a theory for future fast and intrinsic asynchronous computers.

2. REALIZATION, COMPUTATION

It is quite convenient to study concurrency on such abstract levels as state systems and/or transition systems.

Definition 1. An *S system* (state system) A is a tuple $A=(S, \rightarrow)$ of a set S (of states) and a relation $\rightarrow \subseteq S \times S$ (of transitions). A *T system* (transition system) A is a tuple $A=(S, Z, \rightarrow)$ of an S system $(S, \rightarrow)$ and a finite set Z (of names or labels for transitions), such that there exists for any $e \in Z$ a relation $\rightarrow_e \subseteq S \times S$ with

$$\rightarrow = \bigcup_{e \in Z} \rightarrow_e$$

We use the following notation: By induction on $\mathbb{N}$ (nonnegative integers) and Z^*:

$$\forall K \subseteq S\colon \forall s, s' \in S\colon \forall x \in Z^*\colon \forall e \in Z\colon$$

$$s \overset{0}{\underset{K}{\rightarrow}} s' :\Leftrightarrow s = s' \text{ and } s, s' \in K$$

$$s \overset{1}{\underset{K}{\rightarrow}} s' :\Leftrightarrow s \rightarrow s' \text{ and } s, s' \in K$$

$$s \overset{n+1}{\underset{K}{\rightarrow}} s' :\Leftrightarrow \exists s_1 \in K\colon s \overset{n}{\underset{K}{\rightarrow}} s_1 \text{ and } s_1 \overset{1}{\underset{K}{\rightarrow}} s'$$

$$s \underset{K}{\Rightarrow} s' :\Leftrightarrow \exists n \in \mathbb{N}_0\colon s \overset{n}{\underset{K}{\rightarrow}} s'$$

$$s \overset{n}{\rightarrow} s' :\Leftrightarrow s \overset{n}{\underset{S}{\rightarrow}} s'$$

$$s \Rightarrow s' :\Leftrightarrow \exists n \in \mathbb{N}_0\colon s \overset{n}{\rightarrow} s'$$

$$s \rightarrow_\lambda s' :\Leftrightarrow s \overset{0}{\rightarrow} s'$$

$$s \rightarrow_{xe} s' :\Leftrightarrow \exists s_1 \in S\colon s \rightarrow_x s_1 \text{ and } s_1 \rightarrow_e s'$$

$$s \rightarrow_x :\Leftrightarrow \exists s' \in S\colon s \rightarrow_x s'$$

$$s^{\rightarrow} := \{s' \in S\colon s \Rightarrow s'\}$$

$$K^{\rightarrow} := \bigcup_{s \in K} s^{\rightarrow}$$

An S or T system A has a *dimension* $d :\Leftrightarrow \exists$ set S_1: $S \subseteq \mathbb{N}_0^d \times S_1$. Some

coordinate i of $\mathbb{N}_0^d$ of a d-dimensional system A is called

$$k \textit{ safe with respect to } S' \subseteq S : \bigotimes \forall s \in S'^{\frown} : (s)_i \leqslant k$$

$$\textit{safe with respect to } S' : \bigotimes i \text{ is } k \text{ safe for some } k$$

Here $(\ )_i$ denotes the ith coordinate of a vector. A is called (k) *safe* (*with respect to* S') : $\bigotimes$ all coordinates i of $\mathbb{N}_0^d$ of A are (k) safe (with respect to S').

Note that any system A of dimension d is also of any lower dimension as we may redefine the set S_1. However, it will always be obvious from the context what dimension and what set S_1 is actually meant.

Any Petri net, N, is a T system in a canonical sense. Let P_N denote the set of places and E_N the set of events of N. Then $N = (\mathbb{N}_0^d, E_N, \rightarrow)$ of dimension $d = \#P_N$, where $s \rightarrow_e s'$ denotes that in state (marking) s event e can fire and reach the new state s'. In the following $^{\cdot}e$, $e^{\cdot}$, $^{\cdot}p$, and $p^{\cdot}$ denote as usual the sets of places that are connected with the event e by arcs pointing to, or, respectively, from, e and of events that are connected with the place p by arcs pointing to, or, respectively, from, p.

As an example,

$$s \xrightarrow[K]{n}_{x} s' \qquad \text{for } s, s' \in \mathbb{N}_0^d,\ K \subseteq \mathbb{N}_0^d$$

in a Petri net of dimension d means that the "firing sequence" x of length n has fired from marking s to marking s', where all intermediate markings belong to the subset K. Note that here we have combined some of the notation of Definition 1 that have not been stated in this combination but should be self-evident.

With this important model for concurrency in mind we also call the coordinates of a d-dimensional system A the places of A and denote by P_A the set of all places of A. Note that $\#P_A = d$ or $\#P_A = d+1$, depending on whether $S_1 = \varnothing$ or not. We need S_1 as we will also operate with automata-theoretical models where not all components of the states should be interpreted as signals or tokens. We will write a state of a d-dimensional system A as

$$s = (s(p_1), \ldots, s(p_d), s(p_{d+1})) \qquad \text{with } p_i \in P_A \text{ and } s(p_{d+1}) \in S_1 \text{ for}$$

$$S_A = \mathbb{N}_0^d \times S_1$$

We need a few more definitions:

Definition 2. A T system $A=(S, Z, \rightarrow)$ is called

$$\begin{aligned}
&\textit{locally determined}: \Leftrightarrow \forall e\in Z: \forall s, s', s''\in S: (s\rightarrow_e s' \text{ and } s\rightarrow_e s''\succ s'=s'')\\
&\textit{commutative}\quad : \Leftrightarrow \forall e, e'\in Z: \forall s\in S: (s\rightarrow_{ee'} \text{ and } s\rightarrow_{e'e}\succ\\
&\qquad\qquad \exists s'\in S: s\rightarrow_{ee'} s' \text{ and } s\rightarrow_{e'e} s')\\
&\textit{persistent}\quad : \Leftrightarrow \forall e, e'\in Z: \forall s\in S: (e\neq e' \text{ and } s\rightarrow_e \text{ and } s\rightarrow_{e'}\succ\\
&\qquad\qquad s\rightarrow_{ee'})\\
&\textit{confluent}\quad : \Leftrightarrow \forall s, s', s''\in S: (s\Rightarrow s' \text{ and } s\Rightarrow s''\succ \exists s^+\in S:\\
&\qquad\qquad s'\Rightarrow s^+ \text{ and } s''\Rightarrow s^+)\\
&\textit{modular}\quad : \Leftrightarrow A \text{ is persistent and commutative}
\end{aligned}$$

As locally determined and modular systems are confluent—see, e.g., Keller (1975)—and confluency is certainly too restrictive for a general theory of concurrency, modular concurrency must handle locally nondetermined systems. We will introduce two such systems, modular Petri nets and APA nets, in the third section.

Definition 3. An S system $B=(S_B, \rightarrow_B)$ *realizes* an S system $A=(S_A, \rightarrow_A)$ iff: $\exists K$: $S_A\rightarrow \mathcal{P}(S_B)-\varnothing$, $S_A\ni s\mapsto K_s$: $\forall s, s'\in S_A$: $\forall \bar{s}\in K_s$:

(i) $s\rightarrow_A s'\succ \exists \bar{s}'\in K_{s'}$: $\bar{s}\Rightarrow_B \bar{s}'$

(ii) $\exists \bar{s}'\in K_s$: $\bar{s}\Rightarrow_B \bar{s}'\succ s\Rightarrow_A s'$

(iii) $\forall s^0\in S_B$: $\bar{s}\Rightarrow_B s^0\succ \exists s^+\in K^+\big(:=\cup_{s\in S_A} K_s\big)$: $s^0\Rightarrow_B s^+$

B realizes A *promptly* iff in addition there holds

(iv) $\forall s\in S_A$: $\forall \bar{s}\in K_s$: $\exists k\in\mathbb{N}$: $\forall K\subseteq S$: $\forall s^0\in S_B$:

$$\bar{s}\overset{k}{\underset{K}{\rightarrow}} s^0\succ K\cap K_{\delta(s)}\neq\varnothing$$

$$\text{with } K_{\delta(s)}:=\{s^+\in S_B: \exists s'\in S_A: s\Rightarrow_A s' \quad \text{and} \quad s^+\in K_{s'}\}$$

Let us briefly discuss these requirements: By (i) any computation in A can also be fulfilled in B, but may require more steps. All computations of B that start from some state attached to some state of A (i.e., the states of K^+) and (ii) lead to an attached state again can also be done in A, or (iii) that lead to an intermediate state can always be prolonged in B to an attached state in K^+. In other words, B reflects all computations of A (via some coding K of the states) and no hang-ups are introduced due to the process of realization. For a prompt realization (with a slow-down factor $\leqslant k$) it is ensured that after at most k steps of computation in B, starting with a state

$\tilde{s}$ attached to some state s of A, a correct result has been achieved, i.e., B has met a state $s^+ \in K_{\delta(s)}$.

This conception has been developed by the author (Priese, 1980a) independently from other attempts. It is very closely related to the independent definition of a reduction by Kwong (1977), Jensen (1980), and Berthelot, Roucairol, and Valk (1979), and also Kasai and Miller (1979).

This definition states some very general principles for simulation and realization of the computational aspects in various models but does not refer to concurrency itself. For a further theory of concurrent systems we have to specify more realization restrictions. The real difficulty in concurrent, asynchronous systems is the intercommunication of such systems. This general remark includes for example synchronization problems and questions of observability. It seems to me that the handling of interfaces is a good distinction for synchronous and asynchronous systems and should be formalized and added to our realization conception.

Definition 4. An *I/O system* A is a T system $A=(S_A, Z, \rightarrow)$ with some dimension d and two distinguished sets $I_A, O_A \subseteq P_A$ with $I_A \cap O_A = \emptyset$ of *inputs* and *outputs* such that there holds

(i) $S_A \subseteq \mathbb{N}_0^n \times \mathbb{N}_0^m \times \mathbb{N}_0^l \times S_1$, for some set S_1 and $n+m+1=d$ and $\#I_A = n$, $\#O_A = m$

(ii) $\forall s, s' \in S$: $s \rightarrow s' \succ s(p) \geqslant s'(p)\ \forall p \in I_A$, and $s(p) \leqslant s'(p)\ \forall p \in O_A$

An *I/O procedure*, $\mathbb{P}$, for an *I/O* system A is a relation $\mathbb{P} \subseteq S \times S$ with

$$\begin{aligned} \forall s, s' \in S:\ & s \mathbb{P} s' \succ s(p) \leqslant s'(p)\ \forall p \in I_A, \\ & s(p) \geqslant s'(p)\ \forall p \in O_A, \text{ and} \\ & s(p) = s'(p)\ \forall p \in W_A := P_A - (I_A \cup O_A) \end{aligned}$$

The T system $(A, \mathbb{P}) := (S_A, Z, \rightarrow_A \cup \mathbb{P})$ is called the $\mathbb{P}$ *closure* of A. By

$$\overrightarrow{A, \mathbb{P}}$$

we denote the relation $\rightarrow_A \cup \mathbb{P}$ and translate all notations of Definition 1 also for this relation.

The interface of an *I/O* system is given by its inputs and outputs. $s(p)=r$ shall be read as meaning that in state s the input (output, or "inner") wire p (for $p \in I_A$, O_A, or W_A, respectively) carries r signals. An *I/O* system may take off signals from its input wires and send signals on its outputs but not vice versa [see (ii)] and can communicate with its environment via *I/O* procedures that put signals on the inputs and take signals from the outputs of A. When we operate with distributed, asynchronous systems we may regard a module A to be an *I/O* system with an interface

I_A, O_A or we may regard it as a closed system where its communication $\mathbb{P}$ via its interface is corporate with its transitions, $(A, \mathbb{P})$.

Definition 5. An I/O system B *realizes* an I/O system A (*promptly*) iff the properties (i)–(iii) [or (iv), respectively] of Definition 3 are fulfilled and there holds (with the notations of Definition 3) in addition

(0) $\exists \Phi\colon I_A \cup O_A \to \mathfrak{P}(I_B \cup O_B)$:
$\forall p \in I_A\colon \Phi(p) \cap I_B \neq \emptyset$
$\forall p \in O_A\colon \Phi(p) \cap O_B \neq \emptyset$
$\forall p, p' \in I_A \cup O_A\colon p \neq p' \succ \Phi(p) \cap \Phi(p') = \emptyset$
$I_B \cup O_B = \bigcup \Phi(I_A \cup O_A)$

(i) *zero coding*: $\forall p \in O_A\colon \forall s \in S_A\colon \forall \bar{s} \in K_s\colon$
$[s(p) = 0 \Leftrightarrow \forall \bar{p} \in \Phi(p) \cap O_B\colon \bar{s}(\bar{p}) = 0]$

(ii) *monotony*: $\forall s, s' \in S_A\colon \forall p \in I_A \cup O_A\colon$
$s(p) \leqslant s'(p)$ if $p \in I_A$
$s(p) \geqslant s'(p)$ if $p \in O_A$
$s(p') = s'(p')\ \forall p' \in P_A - \{p\}$
$\Big\}$ $\forall \bar{s} \in K_s\colon \exists \bar{s}' \in K_{s'}$ such that there holds:
$\bar{s}(\bar{p}) \leqslant \bar{s}'(\bar{p})\ \forall \bar{p} \in \Phi(p) \cap I_B$,
$\bar{s}(\bar{p}) \geqslant \bar{s}'(\bar{p})\ \forall \bar{p} \in \Phi(p) \cap O_B$, and
$\bar{s}(\bar{p}) = \bar{s}'(\bar{p})\ \forall \bar{p} \in P_B - \Phi(p)$.

(iii) *linearity*: $\forall s_1, s_2 \in S_A\colon \forall \bar{s}_1 \in K_{s_1}\colon \forall \bar{s}_2 \in K_{s_2}\colon \forall p \in I_A \cup O_A\colon$

$$\left(s_3(p')\colon = \begin{cases} s_2(p') & \text{for } p' = p \\ s_3(p') & \text{for } p' \neq p \end{cases} \right), \text{ and } \bar{s}_1(\bar{p}) = \bar{s}_2(\bar{p})\ \forall \bar{p} \in W_B \succ$$

$$\exists \bar{s}_3 \in K_{s_3}\colon \bar{s}_3(\bar{p}) = \begin{cases} \bar{s}_1(\bar{p})\ \forall \bar{p} \in \Phi(p) \\ \bar{s}_2(\bar{p})\ \forall \bar{p} \in P_B - \Phi(p) \end{cases}$$

As several "hand-shaking" procedures operating with ready and acknowledge signals on different wires are quite common in the literature as a coding in realizations, forcing an input place, e.g., to become a pair of an input/output, we cannot demand Φ to be a mapping with $\Phi(I_A) \subseteq I_B$ and $\Phi(O_A) \subseteq O_B$. The property zero-coding is only of technical interest and may be replaced by different properties. Monotony states that whenever we change a state s to a state s' by merely adding some input signals and/or removing some output signals, we can find for any state $\bar{s}$ attached to s a state $\bar{s}'$ attached to s' where we have also only added some input signals and/or removed some output signals. Linearity states that if we can code an input or output p as $\bar{s}_1(p)$ then we can use the same code for a further state $\bar{s}_3$ provided that $\bar{s}_1 = \bar{s}_2$ on W_B. This restriction is important as it ensures that a possible assimilation of input signals in $\bar{s}_1$ has also been done in $\bar{s}_2$. For a further discussion see Priese (1980b).

It should be noted that such requirements are quite natural for concurrent, asynchronous systems. Our realization conception is general enough that most "realization constructions" of the literature remain realization in this formal sense, and on the other hand restrictive enough to allow for strict proofs of impossibility results.

In the sequel we need a simple technical lemma:

Lemma and Definition. For any two I/O systems A and B where B simulates A with a state correspondence K and I/O mapping Φ and for any I/O procedure $\mathbb{P}$ of A we define a relation $\mathbb{P}_{K,\Phi} \subseteq S_B \times S_B$ as: $\forall \bar{s}, \bar{s}' \in S_B$: $\bar{s}\mathbb{P}_{K,\Phi}\bar{s}' : \Leftrightarrow \exists s, s' \in S_A$: $\bar{s} \in K_s$ and $\bar{s}' \in K_{s'}$ and $s\mathbb{P}s'$ and

$$\bar{s}(\bar{p}) \leqslant \bar{s}'(\bar{p}) \; \forall \bar{p} \in I_B$$

$$\bar{s}(\bar{p}) \geqslant \bar{s}'(\bar{p}) \; \forall \bar{p} \in O_B$$

$$\bar{s}(\bar{p}) = \bar{s}'(\bar{p}) \; \forall \bar{p} \in W_B \cup \Phi(\{p \in I_A \cup O_A : s(p) = s'(p)\}).$$

Then there holds

(i) $\mathbb{P}_{K,\Phi}$ is an I/O procedure for B.

(ii) The closed system $(B, \mathbb{P}_{K,\Phi})$ realizes the closed system $(A, \mathbb{P})$ also with the state correspondence K.

3. RESULTS

We will compare Petri nets with modular systems and give some proper hierarchy results.

We need a brief informal outline on APA nets. An (asynchronous, parallel) automaton A is a tuple $A = (I_A, O_A, S_A, R_A)$ of a set I_A of input (wires), O_A of output (wires), S_A of states and a relation $R_A \subseteq (\mathcal{P}(I_A) \times S_A) \times (\mathcal{P}(O_A) \times S_A)$ of transitions. $((M, s), (N, s')) \in R_A$ means that the automaton A in state s may take off one signal from each of its input (wires) of M, switch to state s', and send out one signal on each of its outputs of N. There may be further signals on inputs of $I_A - M$ that will not be changed in the $((M, s), (N, s'))$ transition. An APA net (asynchronous, parallel automata net) over some given automata $A_1, \ldots, A_n$ is a directed graph with copies of $A_1, \ldots, A_n$ in its nodes and connections of inputs and outputs as directed edges with the restriction that any edge connects exactly one output of some copy with one input of the same or another copy of some automata. Such an APA net N is the T system $A = (S_A, Z_A, \rightarrow_A)$, where the states of S describe the distribution of signals (that may accumulate) on wires and local states of the component automata of N, Z_A is the set of all copies of the automata

$A_1,\ldots,A_n$ in N, and for any $B \in Z_A$ the relation $\rightarrow_B$ applies with $s \rightarrow_B s'$ iff in the global state s of N the component B fulfills a local transition with the new resulting global state s' of N. APA $(A_1,\ldots,A_n)$ denotes the class of all APA nets over $A_1,\ldots,A_n$.

It should be noted that APA nets are very closely related with Keller's (1974) speed-independent modules, but APA nets allow for an accumulation of signals on wires. Also, any APA net is an I/O system where I_A, O_A are the sets of inputs, outputs, respectively, of components that are not further connected. One should note that APA nets are by definition modular T systems, as any component can lose its ability of transforming a transition only by making this transition or another of its allowed local transitions (local nondeterminism).

Table I gives examples of automata we will operate with. Figure 1 presents a set of Petri nets that we regard as automata with inputs and outputs and that may be switched together to larger nets according to the construction principle of APA nets. As an example, nets constructed from the components I and J solely will remain free choice Petri nets. By modular Petri nets we denote all Petri nets that are constructed in this sense from these given components. PN(I, J) thus denotes the class of all modular Petri nets constructed from I and J.

Definition 6. Let C_1, C_2 be two classes of I/O systems.

$$C_1 \subseteq C_2 :\Leftrightarrow \forall A \in C_1: \exists B \in C_2: B \text{ realizes } A \text{ promptly}$$
$$C_1 \not\subseteq C_2 :\Leftrightarrow \exists A \in C_1: \forall B \in C_2: B \text{ does not realize } A$$
$$C_1 = C_2 :\Leftrightarrow C_1 \subseteq C_2 \text{ and } C_2 \subseteq C_1$$
$$C_1 \subset C_2 :\Leftrightarrow C_1 \subseteq C_2 \text{ and } C_2 \not\subseteq C_1$$

TABLE I. Examples of Automata*

Name	States	Inputs	Outputs	Transitions	Interpretation
U	a	1,2	3	$1, a \rightarrow 3, a$ $2, a \rightarrow 3, a$	Union of wires
I	a	1	2,3	$1, a \rightarrow 2, a$ $1, a \rightarrow 3, a$	Indeterministic choice
F	a	1	2,3	$1, a \rightarrow \{2,3\}, a$	Fork of a signal
J	a	1,2	3	$\{1,2\}, a \rightarrow 3, a$	Join of two signals
S	a	1,2,3	4,5	$\{1,2\}, a \rightarrow 4, a$ $\{2,3\}, a \rightarrow 5, a$	Simple element
E	a, b	t, s	t^a, t^b, s'	$t, a \rightarrow t^a, a$ $t, b \rightarrow t^b, b$ $s, a \rightarrow s', b$ $s, b \rightarrow s', a$	Kind of a storage element

*Here $s, M \rightarrow s', N$ denotes a transition $((s, M),(s', N)) \in R$.

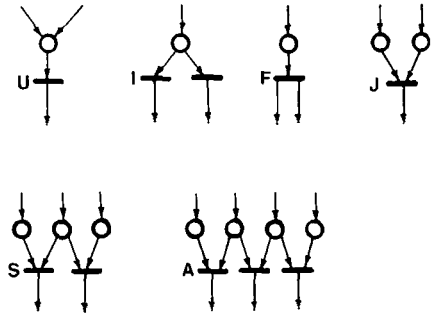

Fig. 1. Modular Petri nets.

Thus for positive results we will require prompt realization but for negative results the absence of any realization. We now can state the main results:

Let GPN denote the class of all generalized Petri nets; RPN the class of all restricted Petri nets; SM the class of all state machines; MG the class of all marked graphs; FC the class of all free choice nets; PFC the class of all pseudo-free-choice nets; SN the class of all simple nets; PN the class of all Petri nets; and APA the class of all APA nets over some finite automata (i.e., with finite sets of states, inputs, and outputs). These definitions are standard in the theory of Petri nets and may be found, for example, in Hack (1973). For any class C of nets we denote by 1-safe C the class of all 1-safe nets in C.

Theorem 1. $\mathrm{GPN} = \mathrm{RPN}$, k-safe $\mathrm{GPN} = 1$-safe $\mathrm{RPN}\ \forall k \in \mathbb{N}$.

It should be noted that this results requires a new proof, as the old constructions in Hack (1975) give only hang-up-free but nonprompt or prompt but not hang-up-free realizations, whereas by Definition 6 our equality ensures hang-up-free *and* prompt realizations. However, such a proof is not too complicated. The next result is just a simple exercise:

Theorem 2.
$$\begin{aligned} \mathrm{SM} &= \mathrm{PN}(U, I) = \mathrm{APA}(U, I) \\ \mathrm{MG} &= \mathrm{PN}(F, J) = \mathrm{APA}(F, J) \\ \mathrm{FC} &= \mathrm{PFC} = \mathrm{PN}(U, I, F, J) = \mathrm{APA}(U, I, F, J) \\ \mathrm{SN} &= \mathrm{PN}(U, F, J, S) = \mathrm{APA}(U, F, J, S) \end{aligned}$$

These classes also define a proper hierarchy modulo nonprompt realization The proof is not trivial and requires some work.

Theorem 3.

$$\begin{matrix} \mathrm{SM} \not\subset \mathrm{MG} \\ \mathrm{MG} \not\subset \mathrm{SM} \end{matrix} \subset \mathrm{FC} = \mathrm{PFC} \subset \mathrm{SN} \subset \mathrm{PN}$$

The following theorem presents some modular decompositions of general PN or APA:

Theorem 4. $\mathrm{PN} = \mathrm{APA} = \mathrm{PN}(U, F, J, A) = \mathrm{APA}(U, F, J, E)$

$$\begin{aligned}\text{1-safe PN} &= \text{1-safe APA} \\ &= \text{1-safe PN}(U, F, J, A) = \text{1-safe APA}(U, F, J, S, E) \\ &= \text{2-safe APA}(U, F, J, E)\end{aligned}$$

If one analyzes the proofs for Theorem 4 it turns out that Petri nets can be promptly realized by modular structures *without* a pairing of wires for hand-shaking procedures with ready and acknowledge signals, and that no signal accumulation in APA nets is needed to realize 1-safe Petri nets.

To show the power of such a formalized approach we will present a classification of modules in PN − SN. We regard the four Petri nets A_0, A_1, A_2, and A_3 of Figure 2. A_0 is a Petri net not in SN ∪ PFC of greatest possible simplicity, as any Petri net in PN − (SN ∪ PFC) must contain A_0 as a subpart. A_3 is Patil's solution of his three-smokers problem (Patil, 1971).

Theorem 5. Not all Petri nets can be promptly realized by some net of $\mathrm{PN}(U, F, J, S, A_1)$. Any Petri net can be nonpromptly realized by some net of $\mathrm{PN}(U, F, J, S, A_0)$.

$$\text{1-safe PN}(U, F, J, S, A_2) \subset \text{1-safe PN} = \text{2-safe PN}(U, F, J, S, A_2)$$

$$\mathrm{PN} = \mathrm{PN}(U, F, J, S, A_3), \quad \text{1-safe PN} = \text{1-safe PN}(U, F, J, S, A_3)$$

As an example we will give an idea of such an impossibility result: In order to prove the first line of theorem 5 we have to find an *invariant* E (a

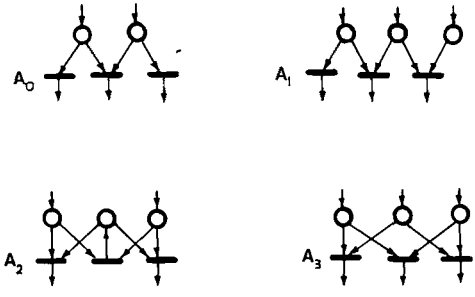

Fig. 2. Nonsimple modular Petri nets "below" A.

property of nets) such that E holds for some Petri net, E is invariant under prompt realization (i.e., if M realizes N promptly and E holds for N, then E also holds for M), but no net of PN(U, F, J, S, A_1) fulfills E. As E we can use the following:

E holds for N : $\bigotimes$

$\exists P \subseteq O_N$: $\exists \mathbb{P}, \mathbb{P}_1$ I/O procedures for N: $\exists s_0 \in S_N$:

(i) $\mathbb{P}_1 \subseteq \mathbb{P}$

(ii) $\forall s' \in S_N$: $\forall p \in P$: $s_0 \underset{N,\mathbb{P}_1}{\rightarrow} s' \succ s'(p) = 0$

(iii) $\forall s' \in S_N$: $s_0 \underset{N,\mathbb{P}_1}{\rightarrow} s' \succ$

$\exists k \in \mathbb{N}$: $\exists \bar{s}, \bar{s}'$: $s' \Rightarrow \bar{s}$ and $\bar{s} \mathbb{P} \bar{s}'$ and $\forall \tilde{s}$:

$\bar{s}' \overset{k}{\rightarrow} \tilde{s} \succ \exists p \in P$: $\tilde{s}(p) > 0$

and $\exists s^+$: $\exists p \in P$: $\bar{s}' \Rightarrow s^+$ and $s^+(p) > 0$

(iv) $\forall s, s' \in S_N$: $\forall p \in I_N \cup O_N$: $s \mathbb{P} s' \succ$

$\exists s_1' \in S_A$: $s \mathbb{P}_1 s_1'$ and $s_1'(p) = s'(p)$

Open Questions:

Let $\{B_1, \ldots, B_n\}$ be a base for serial modules. Is $\{B_1, \ldots, B_n, F, J\}$ then a base for Petri nets?

Serial modules, see Keller (1974), are APA nets with the restriction that at most one signal operates in the whole net. This is equivalent to the concept of normed networks of Rödding and Rödding, see, e.g., Rödding and Rödding (1979). A set M is a base for some class C if $C = \text{APA}(M)$. (*Conjecture*: This is not true.)

Does there exist a decision procedure that tells whether any given set M of automata is a base for serial modules or for Petri nets?

For further impossibility results it may become necessary to state more restrictions on the allowed interface coding Φ—such as the quite reasonable property of "standard coding" in Priese (1980b)—that are true for all constructions of the "positive" realizations results. This leads to the (metamathematical) question, whether there are alternative "smooth" axioms on realizations adequate for asynchronous, concurrent computations with the same hierarchy results.

REFERENCES

Berthelot, G., Roucairol, G., and Valk, R. (1979). "Reductions of Nets and Parallel Programs," in *Net Theory and Applications*, Bauer, W. (Ed.), Lecture Notes in Computer Science 84.

Hack, M. (1973). "Analysis of Production Schemata by Petri-Nets." MAC TR-94, Project MAC, MIT, Cambridge, Massachusetts.

Hack, M. (1975). "Petri Net Languages," Comp. Structure Group Memo 124, Project MAC, Cambridge, Massachusetts.

Hack, M. (1976). "The Equality Problem for Vector Addition Systems is Undecidable," *Theor. Comp. Sci.*, **2**, 77–95.

Jensen, K. (1980). "A Method to Compare the Descriptive Power of Different Types of Petri Nets," in MFCS 1980, *Lecture Notes in Computer Science* Vol. 88.

Kasai, T. and Miller, R. (1979). "Homomorphism between Models of Parallel Computation," IBM RC 7796 (#33742).

Keller, R. (1974). "Towards a Theory of Universal Speed-Independent Modules." *IEEE Trans. Comput.*, **C-23**, 1–33.

Keller, R. (1975). "A Fundamental Theorem of Asynchronous Parallel Computation," in *Parallel Processing*, Feng, T. Y., ed. Springer-Verlag, New York.

Kosaraju, S. (1973). "Limitation of Dijkstra's Semaphore Primitives and Petri Nets," Tech. Rep. 25, Johns Hopkins Univ., Baltimore.

Kwong, Y. S. (1977). "On Reduction of Asynchronous Systems," *Theor. Comput. Sci.*, **5**, 25–50.

Lipton, R. (1976). "The Reachability Problem Requires Exponential Space," Research Report 62, Yale University.

Patil, S. (1971). "Limitations and Capabilities of Dijkstra's Semaphore Primitives for Coordination among Processes," Comp. Struc. Group Memo 57, MIT, Project MAC, Cambridge, Massachusetts.

Priese, L. (1980a). "On the Concept of Simulation in Asynchronous, Concurrent Systems," European Meeting on Cybernetics and Systems Research, Linz 1978; Proceedings in *Progress in Cybernetics and Systems Research*, Vol. II, Hemisphere, Washington, D.C.

Priese, L. (1980b). "An Automatatheoretical Approach to Concurrency," Research Report 12, Series B, Digital Syst. Lab., Helsinki Univ. of Technology.

Rödding, D., and Rödding, W. (1979). "Networks of Finite Automata," European Meeting on Cybernetics and Systems Research, Wien, 1976; in *Progress in Cybernetics and Systems Research*, Hemisphere.