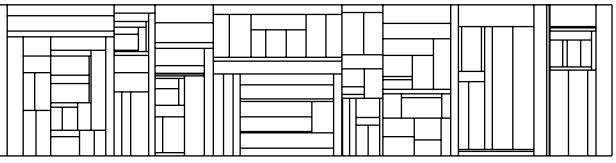
On Decompositions of Stochastic Finite-State Systems

Mogens Nielsen

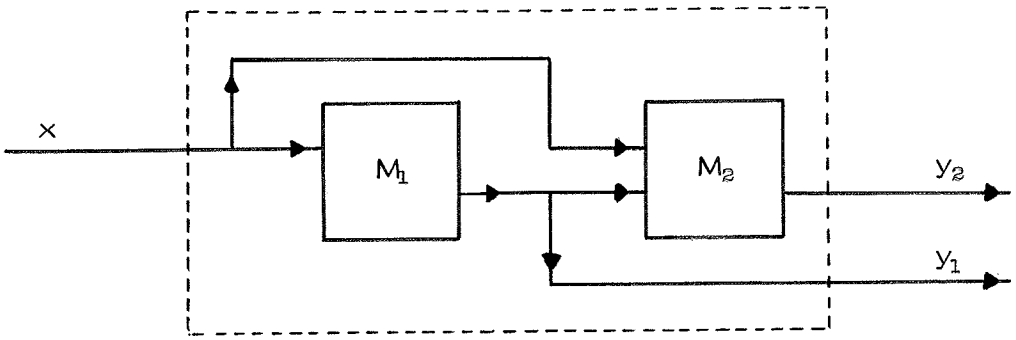
DAIMI PB - 8
February 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF AARHUS
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C, Denmark



ON DECOMPOSITIONS OF

STOCHASTIC FINITE-STATE SYSTEMS



BY

MOGENS NIELSEN

UNIVERSITY OF AARHUS

1973

Contents:

Introduction.

This publication is the thesis for my masters degree, and it is not presented in a form suitable for publishing as a paper. It is preferable if the reader is somewhat familiar with the theory of deterministic and stochastic finite-state systems, but not necessary since the thesis is almost selfcontained.

The main idea of this thesis is the introduction of a new type of information flow in decomposition models for stochastic finite-state systems (the next state of a transition from the front to the tail component of a serial interconnection of two systems). In the loop-free case this phenomenon is introduced in section 2, and its implications are discussed in sections 3-6. In the feed-back case the phenomenon is introduced and shortly discussed in section 7.

Section 1 gives a short introduction to the field of stochastic finite-state systems - for those who are familiar with this field section 1 may be skipped. In section 2 well-known results on serial (cascade) and parallel decompositions of stochastic systems are stated, a new type of decomposition is introduced (denoted strict decomposition), and results on this, equivalent to the above mentioned, are stated. Section 3 is formed as a short comparison of the cascade and the strict decomposition. It turns out that the difference between the two types of decomposition is closely related to the "don't care"-concept, and in connection with the discussion of section 3, the use of "don't care"-transitions in the cascade model with respect to synthesis is pointed out in section 4. In section 5 results on a type of decomposition originally due to E. S. Gelenbe are

stated, and the relation between this, the cascade, and the strict type of decomposition is studied. In section 6 it is indicated (by an example) that the concept of state-splitting might be useful in the strict decomposition model. Finally two feed-back models (maximal and super-maximal) are studied in section 7 - the results on the first due to A. Paz, and the results on the second due to the author.

SECTION 1

Stochastic machines of various types

During the years around 1960 several people from all over the world became interested in the following problem: How to generalize the rapidly growing theory of deterministic automata to a theory of a probabilistic model, (perhaps) better suited to the real world, where many processes can't possibly be described in the strict deterministic or even the indeterministic models of automata theory known at that time and today. One might even argue that, theoretically, no process in the world that we know today could possibly be described in a deterministic model. Anyway, I am not going into any philosophical discussion about the motivation for the work of these people - instead I shall go straight to the point, and define essentially the model that was commonly accepted as the result of their investigations. To do that I shall need the following:

Def. 1.1

A vector $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ with real entries is said to be stochastic iff[1] $\forall i: 0 \leq \pi_i \leq 1$ and [2] $\sum_{i=1}^{n} \pi_i = 1$. A matrix is said to be stochastic iff all it's rows are stochastic vectors.

Notation

Throughout this paper I shall always denote sets (and matrices) in capital letters, and elements from a set (and a matrix) in the corresponding small letters. The i'th element of a set S will sometimes be

referred to as s_i - sometimes just as i. Hopefully this will cause no confusion.

If S is any finite set then $|S|$ denotes the number of elements of S.

Definition 1.2

A stochastic finite state system (SFS) is a system $(S, X, \{M(x)\})$, where S and X are finite, non-empty sets (the set of states and the input-alphabet, respectively), and $\{M(x)\}$ is a set containing $|X|$ stochastic square matrices of order $|S|$ - one for each input symbol.

Interpretation

A SFS may be viewed as a device with an input tape - reading symbols from this tape in discrete time-intervals. At any time-interval the SFS is in one of its states. At the end of each time-interval the SFS reads an input symbol and changes state indeterministically, in that the (i,j)'th entry of $M(x)$, $m_{ij}(x)$, specifies the probability of changing from state s_i to s_j reading the input symbol x. So, the next state of an SFS is dependent only on the present state, the present input symbol, and the probabilities specified in the set of matrices $\{M(x)\}$, which means that the behaviour of the SFS is completely specified from $\{M(x)\}$ - usually called the set of transition matrices, the elements of the matrices called transition probabilities.

Definition 1.3

Let $M=(S, X, \{M(x)\})$ be a SFS. Let X^* denote the set of all finite sequences over the alphabet X, and let $u \in X^*$. If u is equal to the empty

string (usually denoted λ) then M(u) is defined as the identity matrix. If $u = x_1 x_2 \ldots x_k$ then $M(u) = M(x_1) \cdot M(x_2) \cdot \ldots \cdot M(x_k)$.

From simple rules of probability theory and matrix multiplication it follows that [1] M(u) is a stochastic matrix for every $u \in X^*$, and [2] the (i, j)'th entry in M(u) equals the probability of the SFS to be in state s_j when started in s_i and having read the input sequence u (it follows from the interpretation above that the necessary independency among events is assumed).

If all the transition probabilities of a SFS equal either 0 or 1 it is easily seen that the SFS reduces to an ordinary deterministic automaton so def. 1.2 above is really a strict generalization of a deterministic automaton. This imposes among other things that any theorem derivable for SFS's contains a theorem for deterministic automata as a special result.

It is less trivial that the SFS-model is also in some sense a generalization of the classical models of deterministic finite-state automata with output - f. ex. the well known Moore- and Mealy types of machines, concepts with which I expect the reader to be familiar. I shall try to make this clear in the following.

In the literature different probabilistic generalizations of the Moore- and Mealy types of deterministic machines have been given (this is possible since the probabilistic models are more elaborate in structure than the deterministic ones). I shall just give two examples of such generalizations - both of them due to Paz [8].

Definition 1.4

A stochastic Mealy machine is a quadruple $M = (S, X, Y, \{M(y|x)\})$ where S and X are as in def. 1.2, Y is a finite output alphabet, and $\{M(y|x)\}$ is a set containing $|X| \cdot |Y|$ square matrices of order $|S|$, one for each input-output pair of symbols, such that [1] $0 \leq m_{ij}(y|x) \leq 1$ for each $x \in X$, $y \in Y$, $1 \leq i, j \leq |S|$, and [2] $\sum_{y \in Y} \sum_{j=1}^{|S|} m_{ij}(y|x) = 1$, for each $x \in X$, $1 \leq i \leq |S|$.

Interpretation

Our device is now equipped with an output tape. $m_{ij}(y|x)$ is interpreted as follows: Given that the machine is in state s_i and that it reads x on the input tape - $m_{ij}(y|x)$ is the probability that the machine will change state to s_j and at the same time print the symbol y on the output tape.

If an input-output sequence is defined as any pair of sequences from the output and the input alphabet respectively, $(v|u)$, $v \in Y^*$, $u \in X^*$, for which $|v| = |u|$, where $|\cdot|$ denotes the length of a sequence, then the definition of $M(y|x)$ for a Mealy machine can be extended to all input-output sequences in the following way: If $(v|u) = (\lambda, \lambda)$ then $M(v|u) = I$ (the identity matrix of order $|S|$). If $(v|u) = (y_1 y_2 .. y_k | x_1 x_2 .. x_k)$ then $M(v|u) = M(y_1|x_1) \cdot M(y_2|x_2) \cdot .... \cdot M(y_k|x_k)$. It is easily seen that the (i,j)'th entry in $M(v|u)$ is the probability of the machine ending in state s_j printing the sequence v on the output tape, when started in state s_i and fed with the input sequence u.

Definition 1.5

A stochastic Moore machine is a quintuple $M = (S, X, Y, \{M(x)\}, O)$,

where S, X, and Y are as in def. 1.4, $\{M(x)\}$ is a set of square stochastic matrices of order $|S|$, one for each input symbol, and O is a deterministic function from S to Y.

Interpretation

Again (as in def. 1.2) $m_{ij}(x)$ is the probability of a transition from s_i to s_j when x is read on the input tape. If this transition takes place, the Moore machine prints $O(s_j)$ on its output tape.

Notice that there is a slight difference (besides the introduction of probabilistic transitions, of course) between this interpretation and the usual interpretation of a deterministic Moore machine: The output is associated with the next state (the state which the machine is in after having made its transition on an input symbol) and not as usual with the present state (the state that the machine is in before the input symbol is read).

Except for this it is easily seen that both def. 1.4 and def. 1.5 reduce to the corresponding definitions of deterministic machines if all entries in the transition matrices equal 0 or 1.

Definition 1.6

An initiated stochastic system (with or without output - of any type) with state set S is a system together with a fixed $|S|$ - dimensional stochastic row-vector, usually denoted π.

π is interpreted as the initial state distribution of the system. For a SFS $M=(S, X, \{M(x)\})$ and any input sequence $u \in X^*$, $\pi \cdot M(u)$ is a stochastic row-vector specifying the state distribution of M when

started with state distribution π and fed with the input sequence u.

Notation

For a stochastic Mealy or Moore machine M, $p_{\pi}^{M}(v|u)$ denotes the probability of the machine to produce the sequence v on its output tape when started with π as its initial state distribution and fed with the input sequence u. If $\pi = (0, 0, .., 1, .., 0)$ with a single 1 in its i'th place, i.e. the machine starts with probability 1 in state s_i, I shall write $p_{s_i}^{M}$ for p_{π}^{M}. For a stochastic Mealy machine $M = (S, X, Y, \{M(y|x)\})$, $p_{\pi}^{M}(v|u)$ is expressed as $p_{\pi}^{M}(v|u) = \pi \cdot M(v|u) \cdot \eta$, where η is a $|S|$-dimensional column-vector with all entries equal to 1; for a Moore machine the output sequence is directly dependent on the state sequence of the machine, and $p_{\pi}^{M}(v|u)$ is harder to express formally. After a few definitions I shall however, be able to state a certain relationship between the p_{π}^{M}'s of the two types of machines.

Definition 1.7

Two stochastic machines M and M' of either Moore or Mealy type but with the same input and output alphabets are said to be equivalent (state-equivalent) iff for any initial state distribution π (every state s_i) of M there exists an initial state distribution π' (a state s_i') of M' and vice versa, such that $p_{\pi}^{M}(v|u) = p_{\pi'}^{M'}(v|u)$ ($p_{s_i}^{M}(v|u) = p_{s_i'}^{M'}(v|u)$) for every input-output sequence $(v|u)$ of length greater than or equal to 1 (where the length of an input-output sequence is defined as the length of its components).

Remark

The only reason not to consider the empty input-output sequence in the above definition is the fact that the output of a stochastic Moore machine is associated with the next state which means that $p_{\pi}^{M}(\lambda|\lambda) = 0$ for any initiated Moore machine M, while for any initiated Mealy machine M: $p_{\pi}^{M}(\lambda|\lambda) = \pi \cdot M(\lambda|\lambda) \cdot \eta = \pi \cdot I \cdot \eta = 1$.

It is an almost immediate consequence of the definiiton that equivalence is implied by state-equivalence.

Theorem 1.8

For any stochastic Moore machine there exists a state-equivalent stochastic Mealy machine, and vice versa.

proof Let M be any stochastic Moore machine $M = (S, X, Y, \{M(x)\}, O)$. Define a stochastic Mealy machine M' as follows: $M' = (S, X, Y, \{M'(y|x)\})$, where the matrices $M'(y|x) = [m'_{ij}(y|x)]$ is defined by

$$m'_{ij}(y|x) = \begin{cases} m_{ij}(x) & \text{if } O(s_j) = y \\ 0 & \text{otherwise} \end{cases}$$

M and M' are easily seen to be state-equivalent.

The interesting part of the theorem is "and vice versa". Let M now be any stochastic Mealy machine $M = (S, X, Y, \{M(y|x)\})$. Define a stochastic Moore machine $M' = (S', X, Y, \{M(x)\}, O)$, where $S' = Y \times S$, and the matrices $M'(x) = [m'_{ij}(x)]$ are defined as follows ($Y = \{y_1, \ldots, y_m\}$ and the elements of $S' = Y \times S$ are ordered lexicographically):

$$M'(x) = \begin{bmatrix} M(y_1|x) & M(y_2|x) & \ldots & M(y_m|x) \\ M(y_1|x) & \ldots & & \ldots M(y_m|x) \\ \vdots & & & \vdots \\ M(y_1|x) & \ldots & & \ldots M(y_m|x) \end{bmatrix}$$

and finally the output-function O is given by: $O(y_j, s_i) = y_j$, $\forall\, s_i \in S$. Now, first of all the M'-matrices are seen to be stochastic square matrices of order $|S| \times |Y|$ so that M' is a welldefined stochastic Moore machine. Secondly, for any starting state s_i of M and any input-output sequence $(v|u)$ of length greater than or equal to 1 : $p^{M}_{s_i}(v|u) = p^{M'}_{s_i'}(v|u)$ if $s_i' = (y_j, s_i)$ for any $y_j \in Y$. The proof of this equality is very easy and straightforward (induction on the length of the input-output sequences), but it is pretty hard to express formally, so I shall leave it to the reader and refer to [11] for the details. Now, since the equality holds for any y_j it has been shown that M and the constructed M' are state-equivalent, and this completes the proof.

It is to be noted that the proof of theorem 1.8 is based on the fact that the output of a stochastic Moore machine is dependent on the next state of a transition and not on the present state. However, all authors (or at least all that I know of) who have defined the concepts of stochastic Moore and Mealy machines have introduced the same phenomenon. Salomaa [11] defines a weaker generalization of a Mealy machine (requiring the transition and the output to be independent events), and a stronger generalization of a Moore machine (allowing the output-function to be probabilistic - but still a probabilistic function of the next state of a transition); [11] introduces the definition of a Mealy machine used here

as "a finite probabilistic machine", and finally our definition of a Moore machine as "a Rabin machine". Salomaa compares the different types of machines with respect to their ability of defining events on the input alphabet, and he proves a result similiar to theorem 1.8: For any finite stochastic machine there exists an equivalent Rabin machine.

Now, if one compares the definition of a SFS (def. 1.2) with the definition of a Moore machine (def. 1.5) one sees that there is very little difference. As a matter of fact a Moore machine may simply be viewed as a SFS equipped with an output box converting the state of the system to the corresponding O-value. This justifies my previous postulate: The SFS-model is a generalization of the classical models of deterministic finite-state automata with output, and I shall therefore in the forthcomming sections when talking about decomposition of stochastic machines restrict myself to SFS's - with clear conscience. Sometimes I shall assume the SFS to be equipped with an output device (I shall assume it to be a Moore machine), sometimes not. Hopefully it will be clear from the context which is the case.

SECTION 2

Bacon- and strict decomposition

During the sixties the decomposition theory of deterministic sequential machines was studied to a great extent (see f. ex. the book of Hartmanis and Stearns [4] with which I shall assume the reader to be familiar). Now, most of the research in the field of stochastic automata has followed well-explored lines from the theory of deterministic automata (as mentioned a deterministic automaton is a special case of a stochastic one - therefore many of the theorems derived in the theory of stochastic automata are simply generalizations of well-known theorems from the corresponding theory of deterministic automata). So you might think that the above-mentioned decomposition theory was already generalized to a decomposition theory for stochastic machines. This is, however, not the case. A relatively small amount of work has been done to develop this generalized theory. Bacon [1] did some work in 1964, but it wasn't followed up until recently by f. ex. Paz [8] [9], Gelenbe [3], Pugachev [10], Santos [12], (Kuich and Walk [7], Heller [5]). The present work is closely related to [1], [3], [8], and [9], whereas the other references seem to go in other directions.

In this section I shall recall the definitions and results from the paper of Bacon, all of which are of the deterministic-generalizing type, and then introduce a new kind of decomposition for which I shall derive results equivalent to those of Bacon's.

Let me start by recalling some basic definitions of some matrix operations.

Definition 2. 1

Let A and B be any two matrices of dimensions m×n and p×q respectively. The Kronecker product of A and B, $A \otimes B$, is defined as the matrix C of dimension $(m \cdot p) \times (n \cdot q)$ where $[c_{ik,jl}] = [a_{ij} \cdot b_{kl}]$.

In def. 2. 1 the entries of the matrix C are double-indexed. This simply means that the entries are ordered lexicographically. With this made clear it is seen that the Kronecker product of A and B may be thought of as the matrix

$$\begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & \cdots & & \vdots \\ \vdots & & & \\ a_{m1}B & \cdots & & a_{mn}B \end{bmatrix}$$

With this interpretation in mind I shall use the double-indexing of matrices widely throughout this paper.

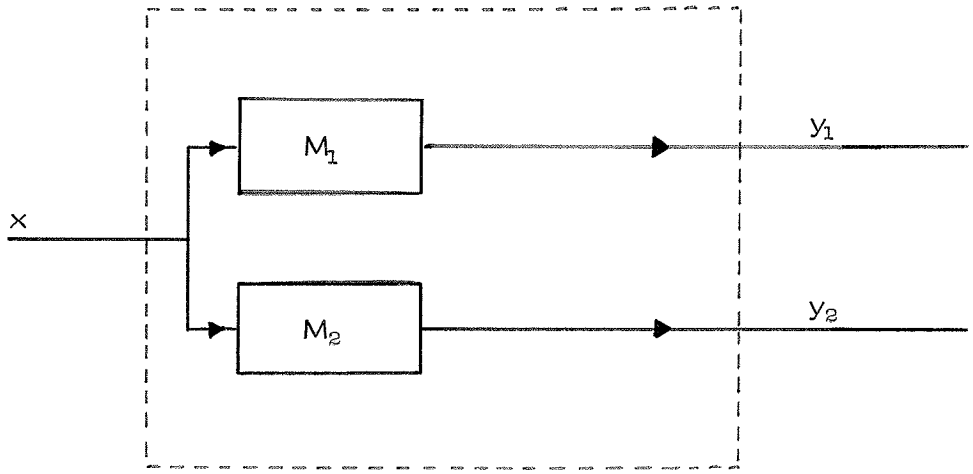
From the definition it is easily seen that if A and B are square, stochastic matrices then $A \otimes B$ is a square, stochastic matrix. This justifies the following definition:

Definition 2. 2

Let $M_1 = (S_1, X, \{M_1(x)\})$ and $M_2 = (S_2, X, \{M_2(x)\})$ be any two SFS's over the same input alphabet. The parallel interconnection of M_1 and M_2, $M_1 \otimes M_2$, is defined as the SFS $M = (S_1 \times S_2, X, \{M(x)\})$, where $M(x) = M_1(x) \otimes M_2(x)$.

Definition 2. 1 and 2. 2 can be extended to define the Kronecker

product (parallel interconnection) of more than two matrices (SFS's). A closer look at def. 2.2 will soon convince you that it is really a generalization of the well-known concept of parallel interconnection of deterministic machines, i.e. $M_1 \otimes M_2$ is a SFS describing what happens if you let M_1 and M_2 work simultaneously in parallel (see fig. 1).



$M_1 \otimes M_2$

Fig. 1

The parallel interconnection of two SFS's.

Definiiton 2.3

Let A be a square, stochastic matrix of order n and let $\{B(i)\}_{i=1}^{n}$ be a set of square, stochastic matrices of order m - one for each row of A. The cascade product of A and $\{B(i)\}$, $A \textcircled{\rightarrow} \{B(i)\}$, is defined as the square, stochastic matrix C of order n·m, where $[c_{ik,jl}] = [a_{ij} \cdot b_{kl}(i)]$.

In this case $A \textcircled{\rightarrow} \{B(i)\}$ may be thought of as the matrix:

$$\begin{bmatrix} a_{11}\ B(1) & a_{12}\ B(1) \ldots\ldots & a_{1n}\ B(1) \\ a_{21}\ B(2) \ldots\ldots & & \vdots \\ \vdots & & \\ a_{n1}\ B(n) \ldots\ldots & & a_{nn}\ B(n) \end{bmatrix}$$

Definition 2. 4

Let $M_1 = (S_1, X, \{M_1(x)\})$ and $M_2 = (S_2, X \times S_1, \{M_2(x, s_1)\})$ be two SFS's. The cascade interconnection of the two, $M_1 \textcircled{\rightarrow} M_2$, is the SFS $M = (S_1 \times S_2, X, \{M(x)\})$, where $M(x) = M_1(x) \textcircled{\rightarrow} \{M_2(x, s_1)\}_{s_1 \in S_1}$. M_1 is called the front- and M_2 the tail-machine of the interconnection.

A graphical representation of a cascade interconnection of two SFS's is shown in fig. 2 - exactly the picture of the well-known cascade interconnection of two deterministic machines. As for def. 2. 2, def. 2. 4 may also easily be extended to define the cascade interconnection of more than two SFS's.

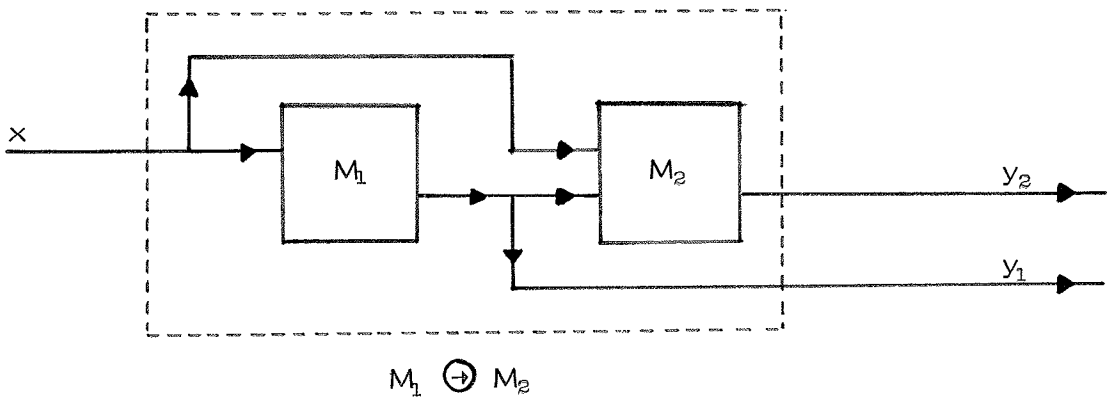


Fig. 2

The cascade interconnection of two SFS's.

Definition 2.5

Let $M=(S, X, \{M(x)\})$ be a SFS. A state s_j is said to be accessible from state s_i iff there exists an input sequence $u \in X^*$ such that $m_{ij}(u) \neq 0$.

Any state is accessible from itself since $m_{ii}(\lambda) = 1$ for every i. Furthermore it is easy to prove the following for any SFS M as in def. 2.5:

$$\Updownarrow \quad \begin{array}{l} s_j \text{ is accessible from } s_i \\ \exists\, u \in X^*,\ |u| \leq |S| : m_{ij}(u) \neq 0 \end{array}$$

From this it follows that you can effectively decide in a finite number of steps which states are accessible from a given state s_i.

Definition 2.6

Let M be a SFS, $M=(S, X, \{M(x)\})$. If $S' \subseteq S$ then the set of states accessible from S' is defined as $\{s_j \mid s_j$ is accessible from some $s_i \in S'\}$.

Definition 2.7

Let M be a SFS as in def. 2.6, and $S' \subseteq S$. S' is said to be a persistent (non-transient) subset of S iff the set of states accessible from S' is a subset of S'.

A "persistent subset S' of S" may be viewed as a "subsystem of the SFS M" in the following sense: delete all rows and coloumns in the transition matrices corresponding to states not in S' - the result is a well-defined SFS with state set S' and input alphabet X. The matrices so constructed are called stochastic submatrices of the transition matrices of M. When I later on refer to a persistent subsystem of a SFS I shall

usually mean a subsystem constructed as described above from a persistent subset of the state set of the SFS.

Definition 2.8

Two SFS's $M=(S, X, \{M(x)\})$ and $M'=(S', X, \{M'(x)\})$ over the same input alphabet are said to be isomorphic iff they are equal up to a permutation and a renaming of state sets, i.e. there exists an isomorphism $\varphi: S \to S'$ such that $m_{ij}(x) = m'_{\varphi(i)\varphi(j)}(x)$, $\forall s_i, s_j \in S$.

Definition 2.9

A SFS M is said to be cascade (parallel) decomposable iff it is isomorphic to a persistent subsystem of a cascade (parallel) interconnection of two SFS's, such that the number of states in each of the components are less than the number of states in M.

From remarks above it follows that def. 2.9 is a generalization of concepts from the theory of deterministic machines. The goal of the next pages is to derive necessary and sufficient conditions for a SFS to be cascade (parallel) decomposable - (hopefully) natural generalizations of the well-known conditions from the deterministic case. But first the following obvious result which I shall need in one of the next sections.

Proposition 2.10

The relation "isomorphic to a persistent subsystem of" is transitive. So is the relation "state equivalent to" between SFS's with output. Furthermore, if M is isomorphic to a persistent subsystem of M', where M and M' are SFS's with output, with isomorphism φ, and

if: $\forall s \in S: O(s) = O'(\varphi(s))$, where S is the state set of M, and O and O' are the outputfunctions of M and M' respectively, then M is state equivalent to a persistent subsystem of M'.

Notation

For any set S, $\mathbb{P}(S)$ denotes the set of all subsets of S. Elements of $\mathbb{P}(S)$ will be denoted P(S). The isomorphism φ in prop. 2.10 may in this notation be expressed as $\varphi: S \to P(S')$, where S' is the state set of M'.

In the following I shall assume the reader to be familiar with the theory of partitions on sets – if he is not, I shall refer to the first section in the book of Hartmanis and Stearns [4]. I shall use the following notation in connection with partitions: A partition π on the set $S = \{s_1, s_2, \ldots, s_n\}$ will be specified by a simple listing of the blocks of π, as indicated in the following specifications of the two trivial partitions $O = \{\overline{s_1}, \overline{s_2}, \ldots, \overline{s_n}\}$ and $I = \{\overline{s_1, s_2, \ldots, s_n}\}$ (the zero- and the identity-partition in the lattice formed by the set of partitions on S, with the usual operators + and •). Formally, $\pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$; The i'th block of π will sometimes be refered to as π_i – sometimes simply as i. If s_i and s_j belong to the same block of π I shall write $s_i \equiv s_j (\pi)$ ($\equiv$ being an equivalence relation).

Definition 2.11

A partition π on the state set of a SFS $M = (S, X, \{M(x)\})$ is said to have the substitution property (S.P.) iff every two states belonging to the same block of π have the same probability of a transition into any

block of π, i.e. if $s_i \equiv s_j\ (\pi)$ then for any block π_l of π and any $x \in X$:

$$\sum_{k \in \pi_l} m_{ik}(x) = \sum_{k \in \pi_l} m_{jk}(x)$$

The definition of S. P.-partitions for deterministic machines is obviously a special case of the above definition, but again since stochastic machines are more elaborate in structure, still other possibilities of generalizing the S. P.-concept might be useful - one of them is mentioned in section 4.

Definition 2.12

A partition π on the state set of a SFS with output $M=(S,X,Y,\{M(x)\},O)$ is said to be output consistent iff the following holds:

$$\forall\, s_i, s_j \in S : s_i \equiv s_j\ (\pi) \Rightarrow O(s_i) = O(s_j)$$

Definition 2.13

Let $M=(S,X,\{M(x)\})$ be a SFS and π a S. P.-partition on S. The π-image of M is a SFS with state set S' equal to the blocks of π, input alphabet X, and transiiton matrices $M'(x)$ defined as follows:

$$\forall x \in X,\ \forall \pi_k, \pi_l \in S' : m'_{\pi_k \pi_l}(x) = \sum_{j \in \pi_l} m_{ij}(x) \quad \text{for some } i \in \pi_k$$

Obviously the π-image of a SFS is a uniquely defined SFS simulating the transitions between blocks of π. If M is a SFS with output and π is a S. P.-partition with output consistence, then it is easily seen that M and its π-image are state equivalent.

Definition 2.14

Two partitions π and τ on the state set of a SFS $M=(S, X, \{M(x)\})$ are said to be independent iff for each pair of blocks (π_k, τ_l) of π and τ respectively for which $\pi_k \cap \tau_l \neq \emptyset$ the following holds:

$$\forall x \in X, \forall s_i \in S: \sum_{j \in \pi_k \cap \tau_l} m_{ij}(x) = \sum_{j \in \pi_k} m_{ij}(x) \cdot \sum_{j \in \tau_l} m_{ij}(x)$$

Now finally after all these definitions I am able to state the following nice and elegant result of Bacon's:

Theorem 2.15

A SFS $M=(S, X, \{M(x)\})$ is cascade decomposable iff there exist two non-trivial partitions π and τ such that

1) π has S.P.

2) $\pi \cdot \tau = O$

3) π and τ are independent

Remark

In the deterministic case conditions 1) and 2) are necessary and sufficient for a cascade decomposition (proved by several authors); condition 3) in theorem 2.15 is, however, easily seen always to be fulfilled in the deterministic case, such that theorem 2.15 really is a nice and elegant generalization of a well-known theorem for deterministic machines.

proof of theorem 2.15:

1) Assume first that M is cascade decomposable into $A \textcircled{\rightarrow} B$, where

$A = (S_A, X, \{A(x)\})$, $B = (S_B, X \times S_A, \{B(x, s_A)\})$ with $|S_A|, |S_B| < |S|$. Then M is isomorphic to a persistent subsystem of $A \oplus B$ (with isomorphism $\varphi : S \to P(S_A \times S_B)$, $\varphi(s) = (\varphi_A(s), \varphi_B(s))$ - which means that M(x) equals a stochastic submatrix of the matrix:

$$(*) \qquad c(x) = [c_{ik,jl}(x)] = [a_{ij}(x) \cdot b_{kl}(x, i)]$$

Summing (*) above over l you get:

$$\sum_l c_{ik,jl}(x) = \sum_l a_{ij}(x) \cdot b_{kl}(x, i) = a_{ij}(x) \cdot \sum_l b_{kl}(x, i) = a_{ij}(x)$$

independently of k. This shows that the partition π on S defined in the following way has S. P.:

$$s_i \equiv s_j \ (\pi) \quad \text{iff} \quad \varphi_A(s_i) = \varphi_A(s_j) \quad \forall s_i, s_j \in S$$

Summing equation (*) over j you get:

$$\sum_j c_{ik,jl}(x) = \sum_j a_{ij}(x) \cdot b_{kl}(x, i) = b_{kl}(x, i) \cdot \sum_j a_{ij}(x) = b_{kl}(x, i)$$

Combining the two summations gives you:

$$(**) \qquad \sum_l c_{ik,jl}(x) \cdot \sum_j c_{ik,jl}(x) = a_{ij}(x) \cdot b_{kl}(x, i) = c_{ik,jl}(x)$$

Now define the partition τ on S:

$$s_i \equiv s_j \ (\tau) \quad \text{iff} \quad \varphi_B(s_i) = \varphi_B(s_j) \quad \forall s_i, s_j \in S.$$

Then, since φ is an isomorphism, $\pi \cdot \tau = 0$ and from (**) above you get immediately that π and τ are independent. (Remember that the $c_{ik,jl}$ - summands which are not in the corresponding sums in the M - matrices are all zero since M(x) is a stochastic submatrix of the C-

matrices). Hence you have that conditions 1), 2) and 3) are fulfilled for the defined partitions π and τ, which both are nontrivial since $|S_A|, |S_B| < |S|$.

2) Now let π and τ be given - non-trivial, satisfying 1), 2) and 3). From π and τ make a reindexing of the states (and the entries of the transition-matrices) of M into double-indices $M(x) = [m_{ik,jl}(x)]$ where i and j ranges over the blocks of π and k, l over the blocks of τ. When this is done define two SFS's $A = (S_A, X, \{A(x)\})$ and $B = (S_B, X \times S_A, \{B(x, s_A)\})$ with state sets equal to the blocks of π and τ respectively - transition matrices of A defined by:

$A(x) = [a_{ij}(x)] = [\sum_l m_{ik,jl}(x)]$ for any τ-block τ_k

which specifies A uniquely according to 1).

The transition matrices of B are defined by:

$B(x, i) = [b_{kl}(x, i)] = [\sum_j m_{ik,jl}(x)]$, whenever this is defined (whenever $\pi_i \cap \tau_k \neq \emptyset$), otherwise the (k, l)'th entry of B(x, i) can be chosen at will (the entries must, however, be chosen such that all B-matrices are stochastic, i.e. such that B becomes a well-defined SFS).

It is easily verified that M is isomorphic to a persistent subsystem of $A \textcircled{\rightarrow} B$ - the subsystem defined by the range of the isomorphism $\varphi: S \to P(S_A \times S_B)$ given by: $\varphi(s) = (\pi_i, \tau_k)$ where $s \in \pi_i \cap \tau_k$ (or equivalently : s is given the index (i, k) in the above mentioned double-indexing). Since both π and τ are non-trivial : $|S_A|, |S_B| < |S|$.

This completes the proof of theorem 2. 15.

Remark

In the construction of the tail machine B in the proof of theorem 2.15 the freedom in specifying some entries in the transition matrices corresponds to the well-known "don't care"-concept from the theory of deterministic machines [4]. In this case you will always get completely unspecified rows, but generally you might give the following definition:

Definition 2.16

A transition matrix of a SFS is said to have don't-care entries iff at least two entries in one of its rows are unspecified. (A single unspecified entry in a row will be determined by the fact that all row-sums equal 1 in a transition matrix).

Example 1

Let $M = (S, X, \{M(x)\})$ be a SFS with $S = \{s_1, s_2, \ldots, s_5\}$, $X = \{a, b\}$ and transition matrices :

$$M(a) = \begin{bmatrix} 1/4 & 0 & 1/4 & 0 & 1/2 \\ 1/4 & 0 & 1/4 & 0 & 1/2 \\ 1/2 & 1/4 & 1/6 & 1/12 & 0 \\ 0 & 3/4 & 0 & 1/4 & 0 \\ 0 & 0 & 2/3 & 1/3 & 0 \end{bmatrix} \qquad M(b) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 1/3 & 1/6 & 1/3 & 1/6 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \end{bmatrix}$$

Now, one can (by simple checking) verify that the conditions of theorem 2.15 hold for the following partitions π and τ on S : $\pi = \{\overline{s_1\, s_2}, \overline{s_3\, s_4}, \overline{s_5}\}$, $\tau = \{\overline{s_1\, s_3\, s_5}, \overline{s_2\, s_4}\}$. From the constructive part of the theorem you get the following cascade decomposition of M:

Front-machine, A:

is simply the π-image of M, i.e. a three state SFS - with one state for each block of π, the same input alphabet as M, and the following transition matrices:

$$A(a) = \begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 3/4 & 1/4 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad A(b) = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

Tail-machine, B:

is a SFS with state set equal to the blocks of τ (a two-state machine), input alphabet $X \times S_A$ where S_A is the blocks of $\pi = \{\pi_1, \pi_2, \pi_3\} = \{\overline{12}, \overline{34}, \overline{5}\}$. Following the proof of theorem 2.15 the transition matrices of B look like the following:

$$B(a,\pi_1) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \qquad B(b,\pi_1) = \begin{bmatrix} 1 & 0 \\ 1/2 & 1/2 \end{bmatrix}$$

$$B(a,\pi_2) = \begin{bmatrix} 2/3 & 1/3 \\ 0 & 1 \end{bmatrix} \qquad B(b,\pi_2) = \begin{bmatrix} 0 & 1 \\ 2/3 & 1/3 \end{bmatrix}$$

$$B(a,\pi_3) = \begin{bmatrix} 2/3 & 1/3 \\ 0 & 1 \end{bmatrix} \qquad B(b,\pi_3) = \begin{bmatrix} 1 & 0 \\ 1/2 & 1/2 \end{bmatrix}$$

Notice that the second rows of the last two matrices ($B(a,\pi_3)$ and $B(b,\pi_3)$) are don't care entries chosen (almost) at will. I shall in one of the next sections return to a discussion of these don't care entries.

For the parallel decomposition a result as nice and elegant as theorem 2. 15 holds:

Theorem 2. 17

A SFS $M = (S, X, \{M(x)\})$ is parallel decomposable iff there exist two non-trivial partitions π and τ on S such that

1) π and τ have S. P.

2) $\pi \cdot \tau = O$

3) π and τ are independent

Remark

Again 1) and 2) are the well-known necessary and sufficient conditions from the theory of deterministic machines.

proof of theorem 2. 17:

The proof follows essentially the same lines as the proof of theorem 2. 15.

1) Suppose that M is parallel decomposable, i. e. isomorphic to a persistent subsystem of $A \otimes B$, where A and B are two SFS's with less states than M, $A = (S_A, X, \{A(x)\})$ and $B = (S_B, X, \{B(x)\})$. Let $\varphi : S \to P(S_A \times S_B)$ be the isomorphism as before, and define partitions π and τ in the same way as in the proof of theorem 2. 15:

$$\left.\begin{aligned} s_i \equiv s_j \ (\pi) \ \text{ iff } \ \varphi_A(s_i) = \varphi_A(s_j) \\ s_i \equiv s_j \ (\tau) \ \text{ iff } \ \varphi_B(s_i) = \varphi_B(s_j) \end{aligned}\right\} \quad \forall\, s_i, s_j \in S$$

The matrices of $A \otimes B$ are from definition given by:

$$C(x) = [c_{ik,jl}(x)] = [a_{ij}(x) \cdot b_{kl}(x)]$$

As before you read directly from this that

$$(*) \quad \sum_l c_{ik,jl}(x) = a_{ij}(x)$$

which shows that π has S. P. - only this time you also have

$$(**) \quad \sum_j c_{ik,jl}(x) = \sum_j a_{ij}(x) \cdot b_{kl}(x) = b_{kl}(x) \cdot \sum_j a_{ij}(x) = b_{kl}(x)$$

which gives you that τ has S. P., and combining (*) and (**):

$$\sum_l c_{ik,jl}(x) \cdot \sum_j c_{ik,jl}(x) = a_{ij}(x) \cdot b_{kl}(x) = c_{ik,jl}(x)$$

which finally gives you that π and τ are independent.

2) Suppose that conditions [1], [2], and [3] hold for partitions π and τ. You can construct two SFS's A and B as in the proof of theorem 2. 15, only this time the entries of the transition matrices of B, $b_{kl}(x, i) = \sum_j m_{ik,jl}(x)$, will be independent of i (since τ has S. P.), so that the cascade products constructed from matrices A(x) and B(x, i) reduce to Kronecker products if you setpossible don't care entries properly, i. e. you have obtained a parallel decomposition.

Example 2

Let M as in example 1 be a 5-state SFS with two input symbols, a and b, - only this time with the following transition matrices:

$$M(a) = \begin{bmatrix} 1/4 & 0 & 1/4 & 0 & 1/2 \\ 1/4 & 0 & 1/4 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \end{bmatrix} \quad M(b) = \begin{bmatrix} 1/2 & 1/4 & 1/6 & 1/12 & 0 \\ 0 & 3/4 & 0 & 1/4 & 0 \\ 1/3 & 1/6 & 1/3 & 1/6 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 1/4 & 1/8 & 5/12 & 5/24 & 0 \end{bmatrix}$$

With the same π- and τ-partitions as in example 1 ($\pi = \{\overline{12}, \overline{34}, \overline{5}\}$ and $\tau = \{\overline{135}, \overline{24}\}$) you easily verify that the conditions of theorem 2. 17 hold,

and the constructive part of the proof gives you the following two components in the parallel decomposition of M:

Machine A

has state set equal to the blocks of π (a three-state machine) and the following transition matrices:

$$A(a) = \begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/2 & 1/2 & 0 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} \qquad A(b) = \begin{bmatrix} 3/4 & 1/4 & 0 \\ 1/2 & 1/2 & 0 \\ 3/8 & 5/8 & 0 \end{bmatrix}$$

Machine B

has state set equal to the blocks of τ (a two-state machine) and transition matrices:

$$B(a) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \qquad B(b) = \begin{bmatrix} 2/3 & 1/3 \\ 0 & 1 \end{bmatrix}$$

Bacon has shown how to combine theorem 2.15 and 2.17 into a theorem stating necessary and sufficient conditions for a SFS to be decomposable into a general loop-free (i.e. a system of parallel and cascade interconnections where for any two components the following holds: if the state of M_i is in the input alphabet of M_j then the state of M_j is not in the input alphabet of M_i) decomposition of more than two SFS's. This is a straightforward result, and I shall not go into any discussion of it here. Instead, I shall return to theorem 2.15 and look upon it in more details.

What does it say, informally? Condition [1] says that a partition, π, must exist such that if you lump the states in the same block of π the result is a well-defined SFS (in the literature [6] you often find the

terminology "π satisfies the condition of lumpability" instead of "π has S. P." and the "π-image" is then called the "lumped system with respect to π"), and condition [3] says that any transition of the given SFS considered as an event can be viewed as two independent transition-events going on at the same time - the first one corresponding to transitions between blocks of π - the second corresponding to transitions in blocks of π (between blocks of τ, the second one knowing which block of π it is to "split").

As mentioned before the cascade model is carried over directly from the theory of deterministic machines. Let's return to section 1 for a moment and view a SFS as a physically working Moore machine (equipped with f. ex. the identity function as its output function). Now, consider the following question: Is the cascade model the most reasonable model for a "serial" interconnection of two or more SFS's? In the theory of deterministic machines things are nice because the cascade model gives you a decomposition of a Moore machine into two physically working Moore machines, making their transitions at the same time. This is possible since the output of a deterministic Moore machine (the front-machine) is dependent on the present state (since the argument may be continued, a cascade interconnection of n Moore machines requires only one time unit to perform the transitions of all its components).

But what about a stochastic Moore machine, for which the output is dependent on the next state of a transition? In the cascade model you can't possibly regard a decomposition of a stochastic Moore machine M into $A \textcircled{\rightarrow} B$ as a decomposition of M into two Moore machines, the first of which making its transitions on the input, and the second making its

transitions on the input and output of the first, because what is transmitted from A to B is the present and not the next state of A .

Don't let me be misunderstood: I don't say that Bacon's decomposition-model is meaningless - but I am saying that it might be more reasonable, since several (all?) papers on stochastic Moore-machines are working with next-state output, to look upon a decomposition-model in which the next-state of the first component-machine is transmitted to the second - i.e. a model in which the interconnection of n Moore-machines would require n time-units to perform all transitions of all components. Working with this model a stochastic Moore-machine would really be decomposed into physically working stochastic Moore-machines.

As a matter of fact Hartmanis and Stearns introduces the idea of this next-state decomposition for deterministic machines in [4], but leaves it immediately with a comment that for greater loop-free interconnections it will lead to some timing problems. Anyway, let's try and see what happens in the simple case of just decomposing a single machine into two.

Definition 2.18.

Let A be a square stochastic matrix of order n and let $\{B(j)\}_{j=1}^{n}$, be a set of square stochastic matrices of order m - one for each coloumn of A . The strict cascade product of A and $\{B(j)\}_{j=1}^{n}$, $A \Phi \{B(j)\}_{j=1}^{n}$ is defined as the square stochastic matrix C of order $m \cdot n$ where $c_{ik,jl} = a_{ij} \cdot b_{kl}(j)$, $1 \leq i, j \leq n$, $1 \leq k, l \leq m$.

Remark

Obviously $A \Phi \{B(j)\}$ may be thought of as the matrix:

$$C = \begin{bmatrix} a_{11} B(1) & a_{12} B(2) & \cdots & a_{1n} B(n) \\ a_{21} B(1) & & & \vdots \\ \vdots & & & \\ a_{n1} B(1) & \cdots & \cdots & a_{nn} B(n) \end{bmatrix}$$

Definition 2. 19.

Let $M_1 = (S_1, X, \{M_1(x)\})$ and $M_2 = (S_2, X \times S_1, \{M_2(x, s_1)\})$ be two SFS's. The strict cascade interconnection of M_1 and M_2, $M_1 \Phi M_2$ is the SFS $M = (S_1 \times S_2, X, \{M(x)\})$ where $M(x) = M_1(x) \Phi \{M_2(x, s_1)\}_{s_1 \in S_1}$.

As definition 2. 4. this definition may be extended to define the strict cascade interconnection of more than two SFS's - and again M_1 (M_2) is called the front (tail) machine.

Definition 2. 20.

A SFS M is said to be strict-decomposable iff it is isomorphic to a persistent subsystem of a strict cascade interconnection of two SFS's each of which have fewer states than M .

Now, let $M = (S_1, X, \{M(x)\})$ be any SFS and π and τ any two partitions on S for which $\pi \cdot \tau = 0$. If all blocks of π (or τ respectively) are of the same size the partition is said to be normal. If either π or τ (or both) are not normal then you may extend the system M introducing one new state for each pair of blocks (π_i, τ_j) for which $\pi_i \cap \tau_j = \emptyset$ and defining the transition-probabilities of these new states as follows:

The transition-probability from any of the original states of M to any of the new is zero and the transition-probability from any of the new states to any state is chosen at will (still in such a way that the extended system is a welldefined SFS). The new SFS obtained in this way is called a normal extension of M with respect to π and τ, and for such a normal extension the following holds: [1] M is isomorphic to a persistent subsystem of it and [2] the partitions π and τ defined on the state-set of it in the straightforward way (the state introduced because $\pi_i \cap \tau_j = \emptyset$ is put in blocks π_i and τ_j) are normal. I shall use the notation $\pi(\tau)$ for both the partition $\pi(\tau)$ itself and for the $\pi(\tau)$-partition on the extended system - defined in this straightforward way.

Definition 2.21.

Let $M = (S, X, \{M(x)\})$ be a SFS and π and τ two partitions for which π has S.P. and $\pi \cdot \tau = 0$ - then π and τ are said to be strictly independent iff there exists a normal extension of M with respect to π and τ such that the following holds for the extended matrices $M'(x)$:

1) $\forall x \in X \quad \forall 1 \le i_o, j_o \le |\pi| \quad \forall 1 \le k_1, k_2, |\tau|:$

$$\sum_l m'_{i_o k_1, j_o l}(x) = \sum_l m'_{i_o k_2, j_o l}(x) \qquad \text{(i.e. } \pi \text{ on } M' \text{ has S.P.)}$$

2) $\forall x \in X \quad \forall 1 \le i_o j_o \le |\pi|, \; 1 \le k_o l_o \le |\tau|:$

$$m'_{i_o k_o, j_o l_o}(x) = \begin{cases} \dfrac{\sum_l m'_{i_o k_o, j_o l}(x) \;\; \sum_i m'_{i k_o, j_o l_o}(x)}{\sum_i \sum_l m'_{i k_o, j_o l}(x)} & \text{iff } \sum_i \sum_l m'_{i k_o, j_o l}(x) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

I am now able to state the following theorem for strict decomposition equivalent to theorem 2.15:

Theorem 2. 22.

A SFS $M = (S, X, \{M(x)\})$ is strictly decomposable iff there exist two nontrivial pratitions π and τ such that

1) π has S. P.

2) $\pi \cdot \tau = 0$

3) π and τ are strictly independent .

proof

1) Assume that M is strictly decomposable into $A \Phi B$, $A = (S_A, X, \{A(x)\})$ and $B = (S_B, X \times S_A, \{B(x, s_A)\})$. The isomorphism, φ , between S and a subset of $S_A \times S_B$ thus given defines, as in the proof of theorem 2. 15, the partitions π and τ on S

$$\forall s_i, s_j \in S: \begin{cases} s_i \equiv s_j \ (\pi) & \text{iff } \varphi_A(s_i) = \varphi_A(s_j) \\ s_i \equiv s_j \ (\tau) & \text{iff } \varphi_B(s_i) = \varphi_B(s_j) \end{cases}$$

As in the above mentioned proof it is easily seen that π has S. P. and that $\pi \cdot \tau = 0$. All that is left to prove is then that π and τ are strictly independent, but this is easy since the transition matrices of $A \Phi B$ (C(x)) defines a natural extension of M with respect to π and τ for which π has also S. P. and

$$\frac{\sum_{l} c_{i_o k_o, j_o l}(x) \cdot \sum_{i} c_{i\, k_o, j_o l_o}(x)}{\sum_{i}\sum_{l} c_{i k_o, j_o l}(x)} =$$

$$\frac{(\sum_{l} a_{i_o j_o}(x) \cdot b_{k_o l}(x, j_o)) \cdot (\sum_{i} a_{i j_o}(x) \cdot b_{k_o l_o}(x, j_o))}{\sum_{i}\sum_{l} a_{i j_o}(x) \cdot b_{k_o l}(x, j_o)} =$$

$$\frac{a_{i_o j_o}(x) \cdot b_{k_o l_o}(x, j_o) \cdot \sum_i a_{i j_o}(x)}{\sum_i a_{i j_o}(x)} = a_{i_o j_o}(x) \cdot b_{k_o l_o}(x, j_o) = c_{i_o k_o, j_o l_o}(x)$$

whenever this is defined ($\sum_i \sum_l c_{i k_o, j_o l}(x) \neq 0$), and obviously $c_{i_o k_o, j_o l_o}(x) = 0$ otherwise, since $c_{i_o k_o, j_o l_o}(x)$ is one of the summands in the double sum. So π and τ are strictly independent.

2) Suppose now that the conditions hold for some partitions π and τ. From the extension given since 3) is fulfilled, the SFS's A and B are constructed with state sets equal to the blocks of π and τ respectively and with transitions defined by (the extended matrix from M(x) is denoted M'(x)):

$$a_{i_o j_o}(x) = \sum_l m'_{i_o k_o, j_o l}(x)$$

for some k_o – no matter which since π has S. P. also on the extended system.

$$b_{k_o l_o}(x, j_o) = \begin{cases} \dfrac{\sum_i m'_{i k_o, j_o l_o}(x)}{\sum_i \sum_l m'_{i k_o, j_o l}(x)} & \text{iff } \sum_i \sum_l m'_{i k_o, j_o l}(x) \neq 0 \\ \text{don't care} & \text{otherwise} \end{cases}$$

With these definitions you get immediately that M' is isomorphic to A ⊙ B and since from definition M is isomorphic to a persistent subsystem of M' it has been proved that M is strictly decomposable by use of proposition 2. 10.

Example 3

Consider the following 5-state SFS with input-alphabet {a, b} and transition matrices:

$$M(a) = \begin{bmatrix} 1/6 & 1/12 & 0 & 1/2 & 1/4 \\ 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 1/3 & 1/6 & 0 & 1/3 & 1/6 \\ 0 & 1/2 & 11/6 & 1/6 & 1/6 \\ 1/4 & 1/8 & 0 & 1/4 & 3/8 \end{bmatrix} \quad M(b) = \begin{bmatrix} 1/3 & 0 & 1/6 & 1/6 & 1/3 \\ 0 & 1/3 & 1/12 & 1/4 & 1/3 \\ 1/2 & 0 & 1/4 & 1/4 & 0 \\ 0 & 1/2 & 1/8 & 3/8 & 0 \\ 1/8 & 0 & 1/3 & 1/3 & 3/24 \end{bmatrix}$$

Choose $\pi = \{\overline{12}\ \overline{34}\ \overline{5}\}$ then π has S.P. and for $\tau = \{\overline{135}\ \overline{24}\}$ $\pi \cdot \tau = 0$. Furthermore there exists a normal extension of M with respect to π and τ (with one new introduced state corresponding to blocks $\overline{5}$ of π and $\overline{24}$ of τ) such that the conditions of definition 2.21 holds, i.e. π and τ are stirctly independent. The matrices in this extended system look like:

$$M'(a) = \begin{bmatrix} & & & & & 0 \\ & & M(a) & & & 0 \\ & & & & & 0 \\ & & & & & 0 \\ & & & & & 0 \\ 0 & 3/8 & 1/8 & 1/8 & 3/8 & 0 \end{bmatrix}$$

$$M'(b) = \begin{bmatrix} & & & & & 0 \\ & & M(b) & & & 0 \\ & & & & & 0 \\ & & & & & 0 \\ & & & & & 0 \\ 0 & 1/8 & 1/6 & 1/2 & 5/24 & 0 \end{bmatrix}$$

Now from theorem 2. 22. it follows that M is strictly decomposable and following the constructive part of the proof you get the following components - completely specified from M' :

Front-machine A

has one state for each block of π and is (as seen before) simply the π-image of M:

$$A(a) = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1/3 & 1/6 \\ 3/8 & 1/4 & 3/8 \end{bmatrix} \qquad A(b) = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 1/2 & 0 \\ 1/8 & 2/3 & 5/24 \end{bmatrix}$$

Tail-machine B

has one state for each block of τ-input alphabet $\{a, b\} \times \{\pi_1, \pi_2, \pi_3\}$ and transition matrices:

$$B(a,\pi_1) = \begin{bmatrix} 2/3 & 1/3 \\ 0 & 1 \end{bmatrix} \qquad B(b,\pi_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$B(a,\pi_2) = \begin{bmatrix} 0 & 1 \\ 1/2 & 1/2 \end{bmatrix} \qquad B(b,\pi_2) = \begin{bmatrix} 1/2 & 1/2 \\ 1/4 & 3/4 \end{bmatrix}$$

$$B(a,\pi_3) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \qquad B(b,\pi_3) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

Theorem 2. 23.

For any given SFS M it is decidable if M is cascade-decomposable and if it is strict-decomposable.

Comment

When I say "given SFS" I mean a complete specification of its transition-matrices with all entries f. ex. rational given as finite fractions as in the examples of this paper. For practical purposes the decidability might give some numerical difficulties, a problem which I shall not go into here.

proof of theorem 2. 23.

For the cascade-decomposition the result is trivial since the conditions from theorem 2. 15 can be directly checked for any pair of state-partitions, and there is, of course, only a finite number of such pairs.

For the strict-decomposition the result is not as trivial. For any chosen non-normal partitions π and τ there is an infinite number of normal extensions, all of which one might think it necessary to consider in order to check if π and τ are strictly independent. This is, however, not so. Pick out any of the entries in the extension of M(x) to be chosen at will, $m^{!}_{i_0 j_0, k_0 l_0}(x)$. Suppose there exists a row corresponding to one of the original states of M, $\pi_{i_1} \cap \tau_{k_0}$, for which $\sum_{l} m_{i_1 k_0, j_0 l}(x) \neq 0$ ($\Rightarrow \sum_{l} m^{!}_{i_1 k_0, j_0 l}(x) \neq 0$) then if π and τ are to be strictly independent for some extension of M with specified $m^{!}_{i_0 k_0, j_0 l_0}(x)$:

$$(*)\quad m!_{i_0 k_0, j_0 l_0}(x) = \frac{\sum_l m!_{i_0 k_0, j_0 l}(x) \cdot \sum_i m!_{i k_0, j_0 l_0}(x)}{\sum_i \sum_l m!_{i k_0, j_0 l}(x)}$$

$$= \frac{\sum_l m!_{i_0 k_0, j_0 l}(x) \cdot m!_{i_1 k_0, j_0 l_0}(x)}{\sum_l m!_{i_1 k_0, j_0 l}(x)}$$

since the specification of definition 2.21. does also hold for $m!_{i_1 k_0, j_0 l_0}(x)$. Now since π is supposed to have S.P. on M' , $\sum_l m!_{i_0 k_0, j_0 l}(x)$ is specified from one of the original rows of M (every block of π is non-empty) and since $\pi_{i_1} \cap \tau_k$ is one of the original states of M, $\frac{m!_{i_1 k_0, j_0 l_0}(x)}{\sum_l m!_{i_1 k_0, j_0 l}(x)}$ is also specified by the given transition-matrices of M. Furthermore, if more than one i_1 exist such that $\sum_l m_{i_1 k_0, j_0 l}(x) \neq 0$ it follows from the fact that the equation of def. 2.21 holds for any entry in the extended system that (*) should specify $m!_{i_0 k_0, j_0 l_0}(x)$ uniquely - no matter which i_1 is chosen. If not, then π and τ are known not to be strictly independent. But otherwise the argument above gives you a unique specification of any entry in the extended system - except for the case where no i_1 exists such that $\sum_l m_{i_1 k_0, j_0 l}(x) \neq 0$, but then the $m!_{i_0 k_0, j_0 l_0}(x)$ - entry has no influence on the question of strict independency (it's specification is then only dependent on other entry-specifications of the same kind - it may simply be set equal to one of the original entries of M: $m_{i_0 k_1, j_0 l_0}(x)$ if then any non-specified entry $m!_{i k_0, j_0 l_0}(x)$ for which $\pi_i \cap \tau_{k_1} \neq \emptyset$ is set equal to $m_{i k_1, j_0 l_0}(x)$).

The result of the arguments above is that only one uniquely determined extension of the transition matrices of M has to be considered to decide the question of strictly independency for any pair of state-partitions π and τ . This proves the theorem.

Remark

For practical purposes you should always start the decidability-algorithm for strict indenpendency by checking if (*) on page 37 holds for the entries in the given matrices - usually this process will tell you immediately that π and τ are not strictly independent and only in a few cases you will have to construct the extension given in the proof of theorem 2. 23. to complete the decidability-algorithm.

The next section will be formed as a discussion of the difference between the cascade-decomposition and the strict-decomposition. But before I start this discussion I want to state two more theorems with relation to the subject.

Theorem 2. 24.

The class of cascade-decomposable SFS's and the class of strict-decomposable SFS's are incomparable.

proof The proof is very easy: By means of the proof of theorem 2. 23. the SFS in example 1 is seen not to be strict-decomposable and in the same manner the SFS in example 3 is seen not to be cascade-decomposable. On the other hand any parallel-decomposition of a SFS may be regarded both as a cascade- and as a strict-decomposition and from example 2 it follows that the two classes have a non-empty intersection.

Theorem 2. 25.

Let $M = (S, X, \{M(x)\})$ be a deterministic finite state system (with transition function $\delta : S \times X \to S$ defined by: $\delta(s_i, x) = s_j$ iff $m_{ij}(x) = 1$) and let π and τ be two partitions on S for which π has S. P. and $\tau \cdot \pi = 0$. Then π and τ are strictly independent iff for every pair of

blocks π_{i_1} and π_{i_2} of π and any $x \in X$ for which $\delta(\pi_{i_1}, x) \subseteq \pi_j$, $\delta(\pi_{i_2}, x) \subseteq \pi_j$ for some block π_j the following holds:

(*) $\forall \tau_k$, $\pi_{i_1} \cap \tau_k \neq \emptyset$, $\pi_{i_2} \cap \tau_k \neq \emptyset$: $\delta(\pi_{i_1} \cap \tau_k, x) = \delta(\pi_{i_2} \cap \tau_k, x)$.

proof Suppose that the condition is fulfilled. Following the proof of theorem 2.23. an extension is easily constructed such that the conditions of definition 2.21. holds. Suppose on the other hand that π and τ are strictly independent, and assume that a τ_k exists such that (*) above does not hold. Let $\delta(\pi_{i_1} \cap \tau_k) = \pi_j \cap \tau_{l_1} \neq \delta(\pi_{i_2} \cap \tau_k) = \pi_j \cap \tau_{l_2}$. Then since π and τ are strictly independent:

$$m_{i_1 k, j l_2}(x) = \frac{\sum_l m!_{i_1 k, j l_1}(x) \quad \sum_i m!_{i k, j l_2}(x)}{\sum_i \sum_l m!_{i k, j l}(x)}$$

but $m_{i_1 k, j l_2}(x)$ and $m_{i_2 k, j l_2}(x)$ contributes to the sums over l and i respectively which means that $m_{i_1 k, j l_2}(x) \neq 0$ - a contradiction.

As mentioned before, the condition of independency is always fulfilled for deterministic transition-matrices. Theorem 2.25. shows that this does not hold for the condition of strict independency. This fact is another indication (besides the one mentioned by Hartmanis and Stearns) of why the strict decomposition has never been looked upon to any greater extent in the theory of deterministic machines. But the fact that the conditions for strict decomposition are more restrictive than the conditions for cascade-decomposition of a deterministic finite-state system is a bad motivation for only considering the cascade-decomposition of stochastic finite-state systems - and as a matter of fact I shall in the next section argue that the situation in the stochastic case is the opposite: Conditions for cascade - are more restrictive than conditions for strict-decomposition.

SECTION 3

A comparison of cascade- and strict-decomposition

If you compare theorem 2.15 with theorem 2.22 you will find them very much alike. Anyway – if you look closer at the proofs of these theorems you will find a small but in some sense essential difference in the two ways of decomposing a SFS. In this section I shall point out this difference and relate it to the concept of don't-care transitions. Don't care transitions appear in the tail machine of the constructed decompositions whenever one of the partitions π or τ is not normal. Let me first look at the phenomenon for the cascade decomposition. Let $M = (S, X, \{M(x)\})$ be a SFS and π and τ two state-partitions satisfying the three conditions of theorem 2.15. From $\pi \cdot \tau = 0$ you get $|S| \leq |\pi| \cdot |\tau| = |S_A| \cdot |S_B|$ where A and B refer to the front- and tailmachine of the decomposition of M with respect to π and τ. If π or τ contains blocks of different sizes you can even conclude: $|S| < |\pi| \cdot |\tau| = |S_A| \cdot |S_B|$. The fact that M is isomorphic to a persistent subsystem of $A \oplus B$ may in the terminology of the last part of the previous section be expressed as follows: The transition matrices of $A \oplus B$, $\{C(x)\}$, are normal extensions of $\{M(x)\}$ with respect to π and τ with $|\pi| \cdot |\tau| - |S|$ introduced don't-care states where:

$$\forall 1 \leq_{i,j} \leq |\pi|, 1 \leq_{k,l} \leq |\tau| :$$

$$c_{ik,jl}(x) = \begin{cases} m_{ik,jl}(x) & \text{if this is defined in } M(x) \\ 0 & \text{if } \pi_i \cap \tau_k \neq \emptyset \text{ but } \pi_j \cap \tau_l = \emptyset \\ \text{don't care} & \text{otherwise} \end{cases}$$

Now, the extended system is naturally (as for strict decomposition) - decomposable with respect to the natural extensions of π and τ on the state set of $A \oplus B$ mentioned before. This means that the conditions of theorem 2.15 hold for the matrices $\{C(x)\}$ - in particular condition [3] holds for any of the introduced zero-entries:

$$\forall_{i_o k_o, j_o l_o},\ \pi_{i_o} \cap \tau_{k_o} \neq \emptyset,\ \pi_{j_o} \cap \tau_{l_o} = \emptyset :$$

$$0 = c_{i_o k_o, j_o l_o}(x) = \sum_l c_{i_o k_o, j_o l}(x) \sum_j c_{i_o k_o, j l_o}(x) =$$

$$\sum_l m_{i_o k_o, j_o l}(x) \sum_j m_{i_o k_o, j l_o}(x)$$

from which you get that either

$$\sum_l m_{i_o k_o, j_o l}(x) = 0 \text{ or } \sum_j m_{i_o k_o, j l_o}(x) = 0$$

(or both).

Consider again π and τ as partitions on S - the state-set of M. For any block of π, π_i, let π_{j_i} be the smallest (or one of the smallest) block of π for which $\sum_l m_{ik, j_i l}(x) \neq 0$ (independent of k since π has S.P.) - from the above equation you now have that $\sum_j m_{ik, jl}(x) = 0$ for any l for which $\pi_{j_i} \cap \tau_l = \emptyset$. From this argument and the corresponding for the partition τ you get:

Theorem 3.1.

Let M = (S, X, {M(x)}) be a SFS, cascade-decomposable with respect to partitions π and τ. For each block π_i of π let π_{j_i} be the

smallest (or one of the smallest) block of π for which $\sum_{l} m_{ik,j_i l}(x) \neq 0$. Then for any block π_j and any block τ_k, $m_{ij,kl}(x) \neq 0$ for no more than $|\pi_{j_i}|$ different l's. For each pair of blocks (π_i, τ_k) let $\tau_{l_{ik}}$ be the smallest (or one of the smallest) block for which $\sum_{j} m_{ik,jl}(x) \neq 0$, then for any block τ_l, $m_{ik,jl} \neq 0$ for no more than $|\tau_{l_{ik}}|$ different j's.

Now, what does this theorem say, intuitively? It says, that if you have a variation in the sizes of the blocks of let's say the partition π corresponding to the front-machine of a cascade-decomposition, then for any transition from block π_i to π_j the number of τ-transitions that can take place at the same time with nonzero probability is limited by the number of states in the smallest block of π which has a non-zero probability of transition from π_i. In another way: If you have a variation in block-sizes then you necessarily must have a certain amount of zeroes in your transition-matrices if your system is to be cascade-decomposable with respect to the given partitions.

Enough about the cascade-decomposition - what about the strict decomposition? In this case it turns out that you have no similar restriction on the transition-matrices of a decomposable SFS. As a matter of fact you can easily construct a strictly decomposable SFS with as much variation in the block sizes of π and τ as you want and without a single zero in the transition matrices of the SFS.

This indicates that the strict decomposition can be used for a greater number of SFS's (see theorem 2.24). The reason for this can be seen if one compares the two extension-processes - one for the cascade-case described above and one for the strict-case described in the previous section.

In the cascade-case you try to extend your transition-matrices in such a way that you obtain a certain proportionality between subrows down rows, but if these subrows are unequally sized you have to fill out with zeroes because the whole thing should remain stochastic, and these two things, the zeroes and the desired proportionality, imposes the zero-structure on the existing transition-matrices. On the other hand you are (almost) completely free in specifying the transition-probabilities from the introduced don't-care states.

In the case of strict decomposition you try to extend your matrices to obtain a proportionality of subrows down columns, and since 1) there is no restriction on the column-sums of a stochastic matrix and 2) the subrows under consideration are equally sized independent of the nature of π and τ, you can effectively make use of the arbitrariness of the transition-probabilities from the don't care states to obtain the desired proportionality and at the same time avoid any imposed zero-structure on the existing matrices. On the other hand the process determines the transition-matrices of both the front and the tail machine completely (except for some very special cases).

Shortly: In the process of strict decomposition you can use the don't-care transitions before the decomposition takes place to make the decomposition possible, and this use will determine the components of the decomposition (almost) completely. In the process of cascade decomposition you can't make any use of the don't-care transitions before the decomposition takes place (i. e. the possibility of the decomposition is independent of how these transitions are specified) but their arbitrariness is carried over to the components (actually the tail-machine) of the decomposition, so that perhaps they may be used to something, whatever that may be (see the next section), after the decomposition has taken place.

I hope that this discussion has justified the proposition that strict-decomposition is applicable to a greater number of SFS's than cascade-decomposition. Notice, however that in the deterministic case this is not true (see theorem 2. 25) since the zero-structure mentioned above is then already in the transition-matrices from definition.

SECTION 4

Don't-care transitions in cascade-decomposition

In the discussion of the last section I mentioned that the don't-care transitions that occur in a cascade decomposition might be used to something. The goal of this section is to find out what.

In the deterministic theory of machines you often relate the use of don't-care transitions to a physical realization (or synthesis - both terms will be used in this section). A similar thing has never been done for stochastic machines - although a theory for synthesis of stochastic machines exists (not developed to any great extent - but anyway!). I shall not go into any deeper discussion of this synthesis theory here - neither shall I start a deeper discussion of the relationship between the decomposition theory and the synthesis theory - although the latter idea would be a very interesting one to do - but I shall introduce just one of the existing synthesis methods for SFS's and relate it to the don't care transitions in a tail-component of a cascade-decomposition of a SFS. The results of this section are not dependent on the chosen synthesis method - similar results would hold for all the methods that I am familiar with (see [8], [2] f. ex.).

The synthesis method that I have chosen rests on the following lemma:

Lemma 4.1.

Let A be a given stochastic matrix of dimension $m \times n$. Then you can effectively construct a set of deterministic (0-1 entries) $m \times n$-matrices, $\{D_i\}_{i=1}^{N}$, with $N \leq m(n-1) + 1$ and a set of non-zero probabilities $\{p_i\}_{i=1}^{N}$ - one for each D_i-matrix - such that

1) $\sum_{i=1}^{N} p_i = 1$

2) $A = \sum_{i=1}^{N} p_i D_i$

proof I claim that the algorithm described in fig. 3 gives you the desired construction of the two sets. It is obvious that all the constructed A_j's are stochastic and that A_{j+1} contains at least one more zero-entry than A_j which gives you that [1] the algorithm stops after a finite number of steps and more specifically [2] the algorithm constructs no more than $m(n-1) + 1$ pairs (D_i, p_i) (since p_j equals one for some $j \leq m(n-1) + 1$). Let N be the number of constructed pairs (N = the final value of i and j in the algorithm) - then from the construction of the matrices A_j you have:

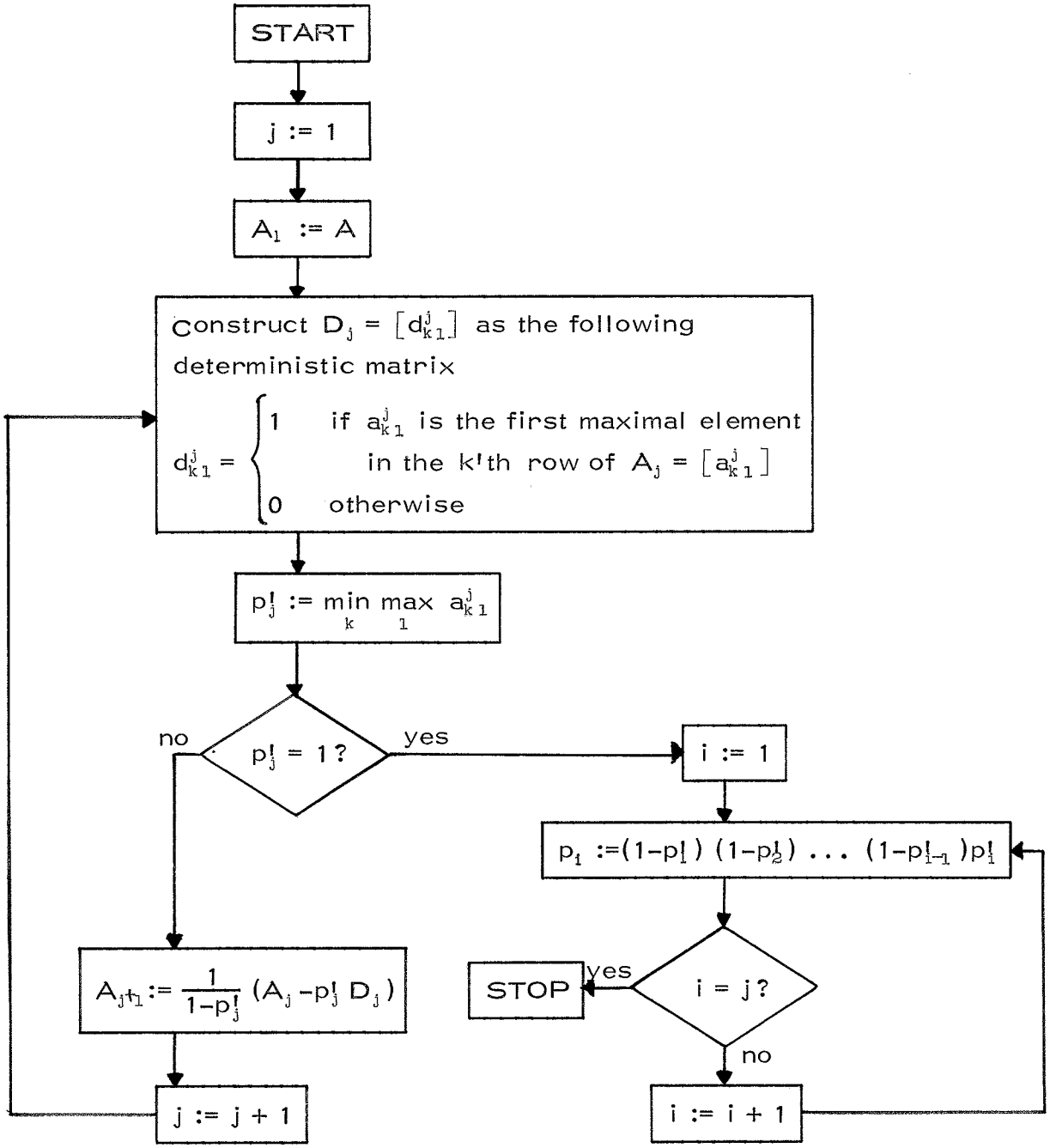


Fig. 3

An algorithm for Lemma 4. 1.

$$
\begin{aligned}
A = A_1 &= p_1' D_1 + (1-p_1') A_2 \\
&= p_1' D_1 + (1-p_1') (p_2' D_2 + (1-p_2') A_3) \\
&\vdots \\
&= p_1' D_1 + (1-p_1')p_2' D_2 + \ldots + (1-p_1')(1-p_2')\ldots(1-p_{N-1}')p_N' D_N \\
&= p_1 D_1 + p_2 D_2 + \ldots + p_N D_N
\end{aligned}
$$

in other words the algorithm gives you A on the desired form as a convex combination of deterministic matrices.

Now, let $M = (S, X, \{M(x)\})$ be any given SFS. Construct the matrix

$$
A_M = \begin{bmatrix} M(x_1) \\ M(x_2) \\ \vdots \\ M(x_p) \end{bmatrix}
$$

where $x_1, x_2, \ldots, x_p$ is a fixed enumeration of the letters from X, i. e. A_M is a stochastic matrix with $|S| \cdot |X|$ rows and $|S|$ columns. If you apply the algorithm described above to this matrix you will get A_M expressed as a convex combination of no more than $|S| \cdot |X| \cdot (|S|-1) + 1 = |X| \cdot |S|^2 - |X| \cdot |S| + 1$ deterministic matrices $A_M = \sum_{i=1}^{N} p_i D_i$. Since the rows of a deterministic matrix correspondto a deterministic transition you can use this result to form a synthesis of M according to fig. 4, in which source is a box that emits

symbols from a finite alphabet $Z = \{z_1, z_2, \ldots, z_N\}$ containing one symbol for each of the deterministic matrices that form A_M above; one symbol is emitted at each timeinterval - the i'th, z_i, with probability p_i. State is a box or a delay keeping the present state of the system, and δ is some combinatorial logic realizing the deterministic transition function $\delta : S \times Z \times X \to S$ given by:

$\delta(s_i, z_u, x_v) = s_j$ iff there is a one in the matrix D_u in the j'th place of the row corresponding to state s_i and input symbol x_v.

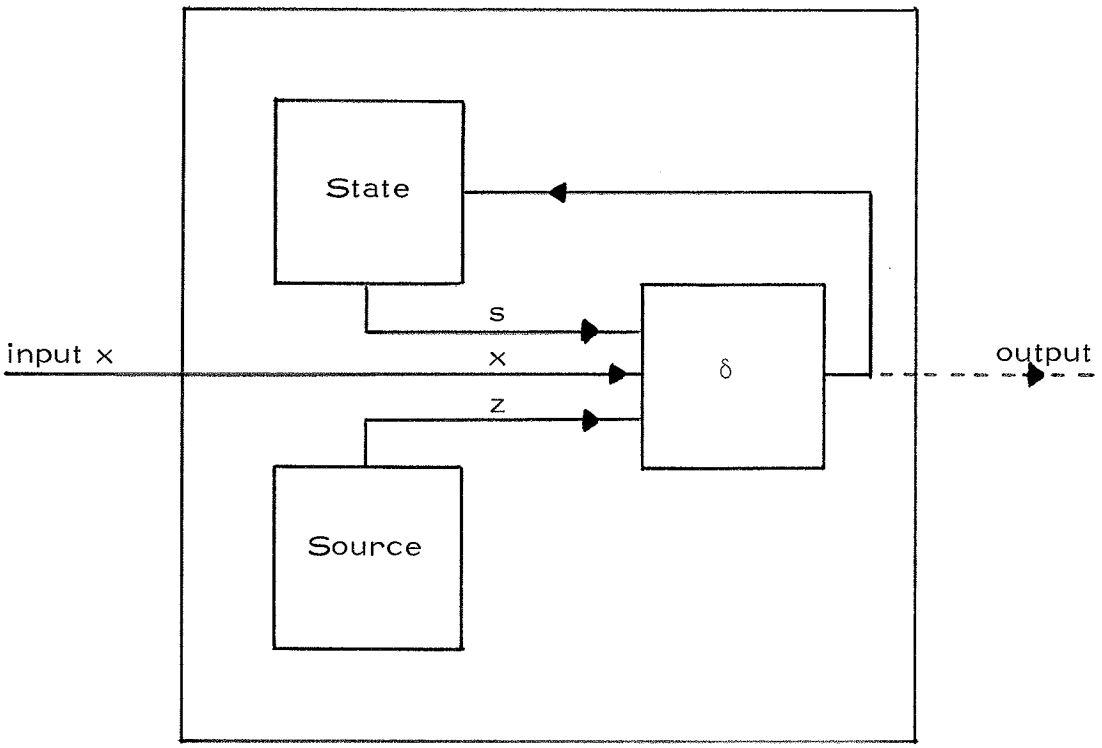


Fig. 4

The synthesis model

I am not going to prove that this model really gives a physical realization of M , i. e. that for each pair of states s_i and s_j and input symbol x_v the probability that the large box of fig. 4 will output (go to) state s_j given that it is in state s_i and receives input-symbol x_v equals the (i, j)'th entry in $M(x_v)$. The proof is very simple and I hope to make it clear by giving an example of the synthesis-process:

Example 4

Let $M = (S, X, \{M(x)\})$ be a two-state SFS with two symbols in its input alphabet $X = \{a, b\}$ and the following transition matrices:

$$M(a) = \begin{bmatrix} 3/4 & 1/4 \\ 1/3 & 2/2 \end{bmatrix} \qquad M(b) = \begin{bmatrix} 1/2 & 1/2 \\ 1/4 & 3/4 \end{bmatrix}.$$

Then

$$A_M = \begin{bmatrix} 3/4 & 1/4 \\ 1/3 & 2/3 \\ 1/2 & 1/2 \\ 1/4 & 3/4 \end{bmatrix}$$

and applying Lemma 4. 1 to this matrix gives you

$$A_M = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{4}\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} + \frac{1}{12}\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Thus the source alphabet consists of four symbols $\{z_1, z_2, z_3, z_4\}$ emitted with probabilities $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{12}\}$ respectively. From the matrices above you can read the following deterministic δ-function

Z	z_1	z_1	z_2	z_2	z_3	z_3	z_4	z_4
X	a	b	a	b	a	b	a	b
s_1	s_1	s_1	s_1	s_2	s_2	s_2	s_2	s_2
s_2	s_2	s_2	s_1	s_1	s_2	s_2	s_1	s_2

which then may be realized by some combinatorial logic forming the box δ in fig. 4 in one of the well-known ways.

This synthesis model gives the possibility of the following alternative definition of a S. P. -partition: A partition π on the state-set of a SFS $M = (S, X, \{M(x)\})$ is said to have S. P. iff for every $x \in X$ and every block π_i there exists a block π_j such that $\forall k \in \pi_i$: $\sum_{l \in \pi_j} m_{kl}(x) = 1$. This definition is also a generalization of the S. P. -concept of deterministic machines, and it has the following nice implication: Any S. P. -partition on the state-set of a SFS will have S. P. in the deterministic sense for the δ-logic in the synthesis of the SFS. In the rest of the paper however the term S. P. -partition will always refer to definition 2. 11.

In the theory of deterministic machines you often measure the complexity of physical realizations of machines. Typical examples of such measurements are 1) the number of internal states and 2) the number of gates in the combinatorial network realizing the transition function of the machine (and the output function if such one exists) or (Hartmanis and Stearns [4]) the number of diodes - i. e. the number of lines going into gates - in this network. The use of many theorems are then illustrated by f. ex. assigning the states or setting don't-care transitions according to some information in the theorems - in order to minimize the complexity of realizations in some sense.

But how do you measure the complexity of a realization of a SFS? It would be reasonable to consider three things : 1) the number of states, 2) the complexity of the deterministic logic realizing the δ-function and 3) the number of symbols in the source-alphabet. Accepting this I shall now mention a few possible ways of using the don't-care transitions that may arise in the tail-component of a cascade-decomposition in order to reduce complexity.

1) Return for a moment to example 1 in section 2. In the decomposition of this example I introduced one don't-care state-corresponding to blocks $\overline{24}$ of τ and $\overline{5}$ of π . This implies that I had a freedom in specifying the transition probabilities from state $\overline{24}$ in the tail machine with respect to inputs $(a, \overline{5})$ and $(b, \overline{5})$. It turns out that I have used this freedom in a very clever way - in that I have specified the transition probabilities such

that I have obtained: $B(a,\overline{5}) = B(a,\overline{34})$ and $B(b,\overline{5}) = B(b,\overline{12})$. In other words: my tail-machine reacts identically to inputs $(a,\overline{5})$ and $(a,\overline{34})$ ($(b,\overline{5})$ and $(b,\overline{12})$), which means that I can reduce the number of inputs to B by 2. This fact has certainly an influence on the complexity of a synthesis of B - with respect to both [2] and [3] in the description above of the complexity of a synthesis of a SFS.

Notice that this use of don't-care transitions corresponds to the one suggested by Hartmanis and Stearns [4] in their short discussion of the special use of the don't-care transitions that appear in the tail-machine of a cascade-decomposition of a deterministic machine. Their book contains a more detailed discussion of the use of don't-care transitions in general.

2) You may also be able to use the don't-care transitions to reduce the number of states in the tail-component. If you are interested in "reading off" from the cascade-decomposition the exact state-behaviour of the decomposed SFS , then you will have to think of the tail machine (and the front)equipped with the identity-function as output function. The definition of state-equivalence (def. 1.7) is then seen to induce the following definition for SFS's:

Definition 4. 2.

Two states of a SFS are said to be equivalent iff the rows corresponding to the two states are equal in every transition matrix.

With this definition El-Ghoroury and Gupta [2] proved the following (for the definition and the use of deterministic state-equivalence, see [4]):

Theorem 4.3.

In a SFS state-equivalence implies and is implied by deterministic state-equivalence in the deterministic machine constructed following the algorithm of synthesis.

The proof is very simple - I am sure that you can easily convince yourself that the theorem holds, so I shall not prove it here, but instead use the result as follows: If, by assigning the transition-probabilities from a don't-care state in the tail-machine properly, you can obtain state-equivalence between this state and some other state(s) - then you will automatically have deterministic state-equivalence between the states in the deterministic machine occuring in the synthesis of the tail-machine.

Example 5

Let M be a 5-state SFS with 1 input symbol and the transition matrix:

$$M(x) = \begin{bmatrix} 3/4 & 0 & 0 & 1/4 & 0 \\ 3/8 & 3/8 & 0 & 1/8 & 1/8 \\ 3/8 & 3/8 & 0 & 1/8 & 1/8 \\ 0 & 1/2 & 1/2 & 0 & 0 \\ 3/4 & 0 & 1/4 & 0 & 0 \end{bmatrix}$$

M is cascade-decomposable with respect to $\pi = \{\overline{123}\ \overline{45}\}$ and $\tau = \{\overline{14},\ \overline{25},\ \overline{3}\}$. The process of decomposition gives you a 2-state one-input front machine with transition matrix

$$A(x) = \begin{bmatrix} 3/4 & 1/4 \\ 1 & 0 \end{bmatrix}$$

and a three-state, two-input ($(x, \overline{123})$ and $(x, \overline{45})$) tail machine with transition matrices

$$B(x, \overline{123}) = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ 1/2 & 1/2 & 0 \end{bmatrix} \quad B(x, \overline{45}) = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 3/4 & 0 & 1/4 \\ - & - & - \end{bmatrix}$$

where - denotes a don't-care entry. State-equivalence in B can be obtained in the following way: Set the last row of $B(x, \overline{45})$ equal to the second row states $\overline{25}$ and $\overline{3}$ of B will then be equivalent.

3) The two possibilities mentioned above suggest the following general way of handling don't-care transitions in the tail-component of a cascade decomposition:

First, observe that non-normal partitions chosen for the decomposition - which is the case I am interested in - give rise to either completely specified or completely unspecified rows in the transition-matrices of the tail-component. In the synthesis-process of this tail component, B, construct the matrix M_B with

rows of dashes inserted for every unspecified row (as in example above), and apply a modified version of the algorithm described in fig. 3 to this matrix - the modifications lying in the loop where the D_j's and the p_j''s are constructed. The modified loop is shown in fig. 5 where the notation $A[k]$ is used for the k'th row of A.

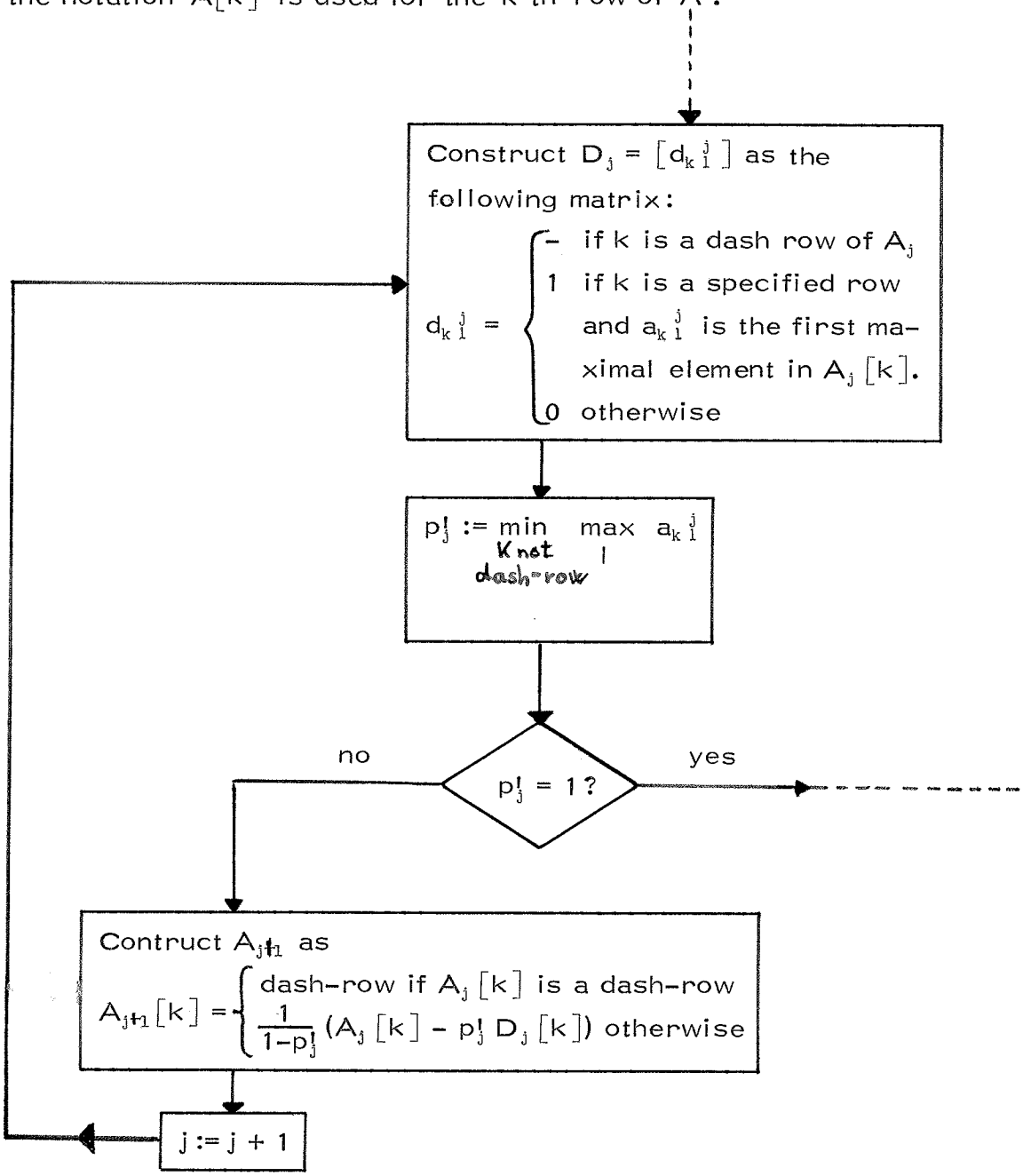


Fig. 5

The modified loop of fig. 3

Following this modified algorithm you will have M_B written as a convex combination of deterministic matrices all of them having dash-rows for any dash-row of M_B - the number of them not greater than $(|S_B| \times |X_B| - d_B)(|S_B| - 1) + 1$ where $|S_B|$ and $|X_B|$ are the number of states and input-symbols of B respectively and d_B is the number of dash rows of M_B.

This algorithm and its natural continuation (the construction of the state-, source- and δ-box), has the following advantages: 1) it certainly minimizes the cardinality of the source alphabet for the synthesis of the tail-machine and 2) it ends up with a deterministic machine in the formed synthesis with one don't-care transition (in the deterministic sense) for each dash-row of M_B, all of which can be treated according to the well-developed theory on the utilization of don't-care transitions of deterministic machines.

As mentioned before I didn't bring up the subject of synthesis with the goal of making a nice and clear theory of the relation between decomposition and synthesis of SFS's - I did it with two purposes: 1) to convince myself and anyone who might read this paper of the possibilities of the until now completely neglected subject and, primarily, 2) to end the discussion of the difference between cascade- and strict-decomposition with the conclusion on page 44 - having proved that the don't-care transitions can be used to something in the cascade-decomposition - not to make the decomposition possible (as in the strict-decomposition) but for ex. to minimize the complexity of a possible physical realization of the decomposed SFS!

SECTION 5

Gelenbe-decomposition

To continue the discussion of the previous section I shall now give a result originally due to E. S. Gelenbe [3], stating something about what happens if a type of decomposition is considered where both the present state and the next state are carried over as information from the front-machine to the tail-machine.

Definition 5. 1.

Let A be a stochastic square matrix of order n and let $\{B(i,j)\}_{i,j=1}^{n}$ be a set of n^2 square stochastic matrices of order m - one for each entry in A . The Gelenbe-product of A and $\{B(i,j)\}$, $A\textcircled{G}\{B(i,j)\}$ is defined as the square stochastic matrix $C = [c_{ik,jl}]$ of order mxn where $[c_{ik,jl}] = [a_{ij} \cdot b_{kl}(i,j)]$.

The definition of a Gelenbe-decomposition of two SFS's follows the old melody of definition 2. 4 -except that the tail component has input-alphabet $X \times S \times S$, - the first S corresponding to the present state of the front component, the second to the next state - that the word "cascade" is replaced with "Gelenbe", and finally "$\textcircled{\rightarrow}$" with "$\textcircled{G}$" . The definition of a Gelenbe-decomposable SFS can be read off definition 2. 9. in the same straightforward way.

I am not going to comment these definitions - I will go directly to the main result of Gelenbe; it is interesting in its form because it is identical to the classical theorem stating necessary and sufficient conditions for cascade-decomposition of deterministic machines:

Theorem 5. 2.

A SFS, $M = (S, X, \{M(x)\})$ is Gelenbe-decomposable iff there exists a non-trivial partition, π, on S with S. P.

proof follows the lines of theorem 2. 15 and 2. 22.

1) Assume that M is Gelenbe-decomposable into $A \textcircled{G} B$ - then M(x) is isomorphic to a stochastic submatrix of $A(x) \textcircled{G} \{B(x, s_i, s_j)\}_{s_i, s_j \in S_A} = [c_{ik,jl}(x)] = [a_{ij}(x) \cdot b_{kl}(x, i, j)]$ with the isomorphism $\varphi : S \to P(S_A \times S_B)$, $\varphi(s) = (S_A, S_B)$. If the entries of M(x) are re-indexed into double-indices according to the isomorphism above, then:

$$\sum_l m_{ik,jl}(x) = \sum_l c_{ik,jl}(x) = \sum_l a_{ij}(x) \cdot b_{kl}(x, i, j) = a_{ij}(x)$$

independently of k which means that π defined as follows has S. P.:

$$\forall s_1, s_2 \in S : s_1 \equiv s_2(\pi) \overset{\text{def}}{\Leftrightarrow} \varphi_A(s_1) = \varphi_A(s_2) .$$

2) Suppose that a non-trivial π with S. P. exists. Construct a partition τ such that $\pi \cdot \tau = 0$ and $1 < |\tau| < |S|$ (this can always be done in a straightforward way). Now the entries of M(x) are reindexed into double-indices according to π and τ (in the same way as usual). The π-image of M - which is to be the front-component of the decomposition -

contains one state for each block of π and the probability of transition from block i to block j on input x equals $\sum_l m_{ik,jl}(x)$. Call this machine $A = (S_A, X, \{A(x)\})$. Construct the tail-machine B as SFS with one state for each block of π, input $X \times S_A \times S_A$ and transition-probabilities:

$$b_{kl}(x,i,j) = \begin{cases} \dfrac{m_{ik,jl}(x)}{\sum_l m_{ik,jl}(x)} & \text{iff } \pi_i \cap \tau_k \neq \emptyset,\ \pi_j \cap \tau_l \neq \emptyset \text{ and } \sum_l m_{ik,jl}(x) \neq 0 \\ 0 & \text{iff } \pi_i \cap \tau_k \neq \emptyset,\ \pi_j \cap \tau_l = \emptyset,\ \sum_l m_{ik,jl}(x) \neq 0 \\ \text{don't care} & \text{otherwise} \end{cases}$$

where "don't care" as usual refers to anything that keeps B stochastic (it will always be possible to assign the don't care entries such that this requirement is met). Obviously, M is isomorphic to a persistent subsystem of $A \textcircled{G} B$ - with isomorphism $\varphi : S \to P(S_A \times S_B)$ such that the φ-image of the h'th state of S equals (i,k) iff $s_h = \pi_i \cap \tau_k$.

Example 6

Let M be a 5-state, 2-input (a and b) SFS with transition matrices:

$$M(a) = \begin{bmatrix} 1/4 & 0 & 3/4 & 0 & 0 \\ 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 1/2 & 0 & 0 & 1/2 \\ 1/4 & 1/4 & 1/3 & 0 & 1/6 \\ 1/3 & 1/6 & 1/4 & 1/4 & 0 \end{bmatrix} \quad M(b) = \begin{bmatrix} 1/6 & 1/6 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 0 & 0 & 2/3 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \end{bmatrix}$$

The partition $\pi = \{\overline{12}\ \overline{345}\}$ is seen to have S. P. and from the constructive part of the proof of theorem 5. 2. you get:

The front-machine:

is a 2-state (s_1, s_2), 2-input (a, b) SFS with transition matrices:

$$A(a) = \begin{bmatrix} 1/4 & 3/4 \\ 1/2 & 1/2 \end{bmatrix} \qquad A(b) = \begin{bmatrix} 1/3 & 2/3 \\ 0 & 1 \end{bmatrix}$$

The tail-machine:

can be constructed f. ex. from the following choice of τ:

$\tau = \{\overline{13},\ \overline{24},\ \overline{5}\}$ - which gives you a 3-state, 8-input

$(\{a, b\} \times \{s_1, s_2\} \times \{s_1, s_2\})$ SFS with transition matrices:

$$B(a, s_1, s_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \underline{0} & \underline{1} & \underline{0} \end{bmatrix} \quad B(a, s_1, s_2) = \begin{bmatrix} 1 & 0 & 0 \\ 1/3 & 1/3 & 1/3 \\ \underline{0} & \underline{0} & \underline{1} \end{bmatrix}$$

$$B(a, s_2, s_1) = \begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 1/2 & 0 \\ 2/3 & 1/3 & 0 \end{bmatrix} \quad B(a, s_2, s_2) = \begin{bmatrix} 0 & 0 & 1 \\ 2/3 & 0 & 1/3 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

$$B(b, s_1, s_1) = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1 & 0 & 0 \\ \underline{0} & \underline{0} & \underline{1} \end{bmatrix} \quad B(b, s_1, s_2) = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 0 & 0 & 1 \\ \underline{0} & \underline{0} & \underline{1} \end{bmatrix}$$

$$B(b_{\cdot}, s_2, s_1) = \begin{bmatrix} \underline{1} & \underline{0} & \underline{0} \\ \underline{0} & \underline{1} & \underline{0} \\ \underline{0} & \underline{0} & \underline{1} \end{bmatrix} \quad B(b_{\cdot}, s_2, s_2) = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

where all underlined entries are don't-care entries chosen at will.

In his paper [3] Gelenbe also tries to prove

Proposition 5.3.

Let $M = (S, X, \{M(x)\})$ be a SFS, Gelenbe-decomposable with respect to partitions π and τ into $A \textcircled{G} B$. Then π and τ are independent iff

$$(*) \qquad \forall s_i, s_{j_1}, s_{j_2} \in S_A \ \forall x \in X : B(x, i, j_1) = B(x, i, j_2) .$$

Unfortunately, Gelenbe almost overlooks the case in which π and τ contain blocks of different sizes - both in the proof of theorem 5.2. and in the proof of this proposition. The proof of theorem 5.2. is quite easy to repair - but the proposition needs a few remarks. F. ex., it is not at all obvious what the equal-sign between the two stochastic matrices in (*) is supposed to mean (remember that both matrices are probably only partly specified.). If the equal-sign is supposed to mean an ordinary equal-sign between any two complete specifications of the two matrices, then the proposition is false. This fact is quite trivial - I am, however, going to show it in the following example:

Example 7:

Let M be a 5-state, 1-input (a) SFS with transition matrix:

$$M(a) = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 3/4 & 1/4 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$\pi = \{\overline{12}, \overline{345}\}$ has S. P. and for $\tau = \{\overline{13}, \overline{24}, \overline{5}\}$ you get the following Gelenbe decomposition of M:

Front-machine A:

has two states, one input symbol.

$$A(a) = \begin{bmatrix} 1/2 & 1/2 \\ 0 & 1 \end{bmatrix}$$

Tail-machine B:

has 3 states, 4 inputsymbols.

$$B(a, s_1, s_1) = \begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 1/2 & 0 \\ \underline{1} & \underline{0} & \underline{0} \end{bmatrix} \quad B(a, s_1, s_2) = \begin{bmatrix} 0 & 1 & 0 \\ 1/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$B(a, s_2, s_1) = \begin{bmatrix} \underline{1} & \underline{0} & \underline{0} \\ \underline{0} & \underline{1} & \underline{0} \\ \underline{0} & \underline{0} & \underline{1} \end{bmatrix} \quad B(a, s_2, s_2) = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 3/4 & 1/4 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

This example shows that the proposition is false with the interpretation of the equal-sign suggested above, in that π and τ are independent but $B(a, s_1, s_1) \neq B(a, s_1, s_2)$ and $B(a, s_2, s_1) \neq B(a, s_2, s_2)$. The first inequality shows that the propostion does not allow two don't-care entries to be compared, the second that it does not even allow a don't-care entry to be compared with an entry specified from the original system. This leads to the following interpretation of the equal-sign in (*):

Definition 5. 4.

Two matrices with don't-care entries A and B are said to be equal, $A = B$, iff the (i, j)'th entry of A equals the (i, j)'th entry of B for every (i, j) for which neither one nor the other entry is a don't-care entry.

Theorem 5. 5.

With the interpretation given in definition 5. 4. of the equal sign in (*), proposition 5. 3. holds true.

A proof of theorem 5. 5. is obtained from Gelenbe's proof in a straightforward way. I am not going to do it here - instead, I shall prove the following:

Theorem 5.6.

Let M be a SFS, $M = (S, X, \{M(x)\})$, Gelenbe-decomposable into $A \textcircled{G} B$ with respect to partitions π and τ. Then π and τ are strictly independent iff

$$\forall s_{i_1}, s_{i_2}, s_j \in S_A \forall x \in X: B(x, s_{i_1}, s_j) = B(x, s_{i_2}, s_j)$$

where the equal-sign is defined in definition 5.4.

proof

1) First assume that π and τ are strictly independent. Then there exists a normal extension with respect to π and τ satisfying the requirements from definition 2.21 Now let $b_{k_o l_o}(x, i_1, j_o)$ be any care (not a don't care) entry in $B(s, i_1, j_o)$. Then either $\pi_{i_1} \cap \tau_{k_o} \neq \emptyset$, $\pi_{j_o} \cap \tau_{l_o} = \emptyset$ and $\sum_1 m_{i_1 k_o j_o l}(x) \neq 0$, in which case $b_{k_o l_o}(x, i_1, j_o)$ equals 0 and for any other i $b_{k_o l_o}(x, i, j_o)$ is either a zero entry or a don't-care entry, or in the other case $\pi_{i_1} \cap \tau_{k_o} \neq \emptyset$, $\pi_{j_o} \cap \tau_{k_o} \neq \emptyset$ and $\sum_1 m_{i_1 k_o, j_o 1}(x) \neq 0$:

$$b_{k_o l_o}(x, i_1, j_o) = \frac{m_{i_1 k_o, j_o l_o}(x)}{\sum_1 m_{i_1 k_o, j_o 1}(x)} = \frac{m'_{i_1 k_o, j_o l_o}(x)}{\sum_1 m'_{i_1 k_o, j_o 1}(x)} = \frac{\sum_i m'_{i k_o, j_o l_o}(x)}{\sum_i \sum_1 m'_{i k_o, j_o 1}(x)}$$

where the primes refer to the extended system. In the last case $b_{k_o l_o}(x, i_1, j_o)$ is written as an expression independent of i_1, which means that for any $s_{i_2} \in S_A$ for which $b_{k_o l_o}(x, i_2, j_o)$ is a care entry $b_{k_o l_o}(x, i_1, j_o) = b_{k_o l_o}(x, i_2, j_o)$ - and the first part of the theorem has been proved.

2) And now to the if-part of the theorem. From the given decomposition of the SFS M, it is possible to construct another one with the same π- and τ-partitions, but with a modified tail-component in which the equality of the transition-matrices holds with the usual interpretation of equality between matrices (strict equality between any two entries). This is an easy matter. First notice that don't-care entries in the tail-component appear in full rows. If the k'th row of $B(x, i, j)$ is a don't-care row, then it is replaced by the k'th row of $B(x, i_1, j)$ for any i_1 in which the k'th row is a care-row (if the k'th row is a care-row for more than one i_1 they will all be equal from assumption). If no such i_1 exists then the k'th row of all matrices $\{B(x, i, j)\}_{s_i \in S_A}$ is replaced with one fixed stochastic $|\tau|$-dimensional vector. For this modified tail-component $B' = (S_{B'}, X \times S_A \times S_A, \{B'(x, s_A, s_A)\})$ the following holds:

$$\forall x \in X \; \forall s_{i_1}, s_{i_2}, s_j \in S_A : B'(x, s_{i_1}, s_j) = B'(x, s_{i_2}, s_j)$$

with the strict interpretation of the equal sign. The Gelenbe-interconnection of A and B', C, defines a normal extension of M with respect to π and τ on which the extended π-partition has S. P. and for which

$$\forall x \in X \; \forall 1 \leq i_o, j_o \leq |\pi| \; \forall 1 \leq k_o, l_o \leq |\tau| : \frac{\sum_l c_{i_o k_o, j_o l}(x) \cdot \sum_i c_{i k_o, j_o l_o}(x)}{\sum\sum_{i\,l} c_{i k_o, j_o l}(x)} =$$

$$\frac{(\sum_l a_{i_o j_o}(x) \cdot b_{k_o l}(x, i_o, j_o)) \cdot (\sum_i a_{i j_o}(x) \cdot b_{k_o l_o}(x, i, j_o))}{\sum\sum_{i\,l} a_{i j_o}(x) \cdot b_{k_o l}(x, i, j_o)} = \frac{a_{i_o j_o}(x) \cdot (b_{k_o l_o}(x, i_o, j_o) \cdot \sum_i a_{i\, i_o}(x))}{\sum_i a_{i j_o}(x)} =$$

$a_{i_o j_o}(x) \cdot b_{k_o l_o}(x, i_o, j_o) = c_{i_o k_o, j_o l_o}(x)$

if $\sum\sum_{i\,l} c_{ik_o, j_o l}(x) \neq 0$ and $c_{i_o k_o, j_o l_o}(x) = 0$ otherwise. So, the normal extension satisfies the requirements of definition 2.21 and π and τ are thus strictly independent.

From theorem 5.5. and 5.6. it follows that Gelenbe-decomposition subsumes both cascade- and strict-decomposition. One might have illustrated the difference between these two decompositions from their relationships to Gelenbe-decomposition given in the two theorems - obviously one would have reached the same conclusion as the one on page 44.

SECTION 6

State-splitting

Paz [8] has shown in some examples that it might be useful to introduce a theory of state-splitting in connection with the cascade-decomposition of SFS's, and recently Santos [12] did some rather general work on the idea. I shall not go into the work of Santos in any detail - I shall just show in one single example that the state-splitting idea can also be used in relation with the strict-decomposition.

$$M(a) = \begin{bmatrix} 1/24 & 13/24 & 5/12 \\ 1/12 & 7/12 & 1/3 \\ 1/6 & 17/30 & 4/15 \end{bmatrix}$$

M is neither cascade- nor strict-decomposable (since none of the three non-trivial partitions on the state set of M has S. P., M is not even Gelenbe-decomposable). To make the system decomposable, however, Paz suggests to split one of the states (s_i) into two states in the following way: Duplicate the i'th row of M(a) and then divide the i'th column into two columns whose sum equals the original column. You have then obtained a 4-state system, which is state-equivalent to the original 3-state system, if the outputs from the two states introduced by the splitting are identified. (Notice that the output function O of a Moore-machine may simply be viewed as a partition ω on the state set of the machine given by: $s_i \equiv s_j$ (ω) iff $O(s_i) = O(s_j)$. In this terminology the two states introduced by the splitting are put into the same block of the output-partition ω). It is to be noted that any of

the existing reduction-algorithms for stochastic systems will reduce the 4-state system to the original 3-state system immediately, but the idea is, of course, to perform the state-splitting in such a way that the 4-state system is decomposable, i. e. if f. ex. the second state of the above system is chosen to be split then to determine the a_{ij}'s in the matrix

$$M'(a) = \begin{bmatrix} 1/24 & a_{12} & a_{13} & 5/12 \\ 1/12 & a_{22} & a_{23} & 1/3 \\ 1/12 & a_{32} & a_{33} & 1/3 \\ 1/6 & a_{42} & a_{43} & 4/15 \end{bmatrix}$$

in such a way that [1] the sum of the a-columns equals the column obtained by duplicating the second row in M(a) , [2] there exists a non-trivial S. P. -partition π on the state-set of M' (say $\{\overline{12}\ \overline{34}\}$) and [3] there exists another non-trivial partition τ (say $\{\overline{13}\ \overline{24}\}$) such that $\pi \cdot \tau = 0$ and π and τ are independent.

It is easily seen that with the chosen splitting-state and the suggested partitions π and τ there is no solution for the a_{ij}'s satisfying the requirements above. But if the term "independent" is replaced with "strictly independent" then the following matrix will be a solution:

$$M''(a) = \begin{bmatrix} 1/24 & 3/24 & 5/12 & 5/12 \\ 1/12 & 1/12 & 1/2 & 1/3 \\ 1/12 & 1/4 & 1/3 & 1/3 \\ 1/6 & 1/6 & 2/5 & 4/15 \end{bmatrix}$$

Since this system from construction fulfills the requirement of the strict-decomposition with respect to π and τ, the result of the above considerations is that M is state-equivalent to (in general a persistent subsystem of) the strict interconnection of the following two SFS's:

Front-machine:

is a two-state (s_1, s_2), one input (a) SFS with transition-matrix:

$$A(a) = \begin{bmatrix} 1/6 & 5/6 \\ 1/3 & 2/3 \end{bmatrix}$$

Tail-machine:

is a two-state, two-input ($\{a\} \times \{s_1, s_2\}$) SFS with transition-matrices:

$$B(a, s_1) = \begin{bmatrix} 1/4 & 3/4 \\ 1/2 & 1/2 \end{bmatrix} \qquad B(a, s_2) = \begin{bmatrix} 1/2 & 1/2 \\ 3/5 & 2/5 \end{bmatrix}$$

SECTION 7

MAXIMAL AND SUPERMAXIMAL INTERCONNECTIONS

Until today the nicest result obtained in the decomposition theory of deterministic machines is the classical Krohn-Rhodes theorem, stating that every deterministic system has a loop-free decomposistion in which each component is either a simple group accumulator or a 2-state reset-machine. I shall not explain this theorem further. I assume the reader to be familiar with it - if he is not, I am convinced that he understands the flavour of the theorem, anyway, from the explanation above: Every deterministic machine can be loop-free decomposed into some very simple components. Unfortunately both the theorem itself and the proof of it are very algebraic oriented - working not with machines as introduced in this paper but with their semi-groups, and this algebraic characterisation of machines does not carry over to stochastic machines directly. As a matter of fact it is still an open question whether a result similar to the Krohn-Rhodes theorem holds for stochastic machines.

If, however, you don't restrict yourself to loop-free decompositions, the situation is quite different. Paz [9] has been working on the problem for the following kind of decomposition:

Definition 7.1.

Let $A = (S_A, X \times S_B, \{A(x, s_B)\})$ and $B = (S_B, X \times S_A, \{B(x, s_A)\})$ be two SFS's. The maximal-interconnection of A and B is defined as the system $C = (S_A \times S_B, X, \{C(x)\})$ where

$$C(x) = [c_{ik,jl}(x)] = [a_{ij}(x,k) \cdot b_{kl}(x,i)] .$$

It is easily checked that the maximal interconnection of two SFS's is a well-defined SFS and that the definition may be extended to a definition of maximal interconnection of more than two SFS's, $M_1, M_2, \ldots, M_n$ - the i'th of which having state set S_i and input alphabet $X \times S_1 \times S_2 \times \ldots \times S_{i-1} \times S_{i+1} \times \ldots \times S_n$. The definition of a maximal decomposable SFS can be read off definition 2.9. in the straightforward way.

The interpretation of a maximal interconnection is as follows: While in the cascade interconnection of n SFS's $M_1, M_2, \ldots, M_n$, a transition in M_i is dependent only on the present state of all M_j's for which $j \leq i$, in a maximal interconnection a transition of M_i is dependent on the present state of all the M_j's, $1 \leq j \leq n$. As for the cascade-interconnection, a maximal interconnection of n SFS's requires only one timeinterval to make all transitions of all components on an input-symbol, and from the interpretation it is easily seen that the cascade-decomposition is a special case of maximal interconnection.

A graphical illustration of a maximal interconnection of 2 SFS's is shown in fig. 6

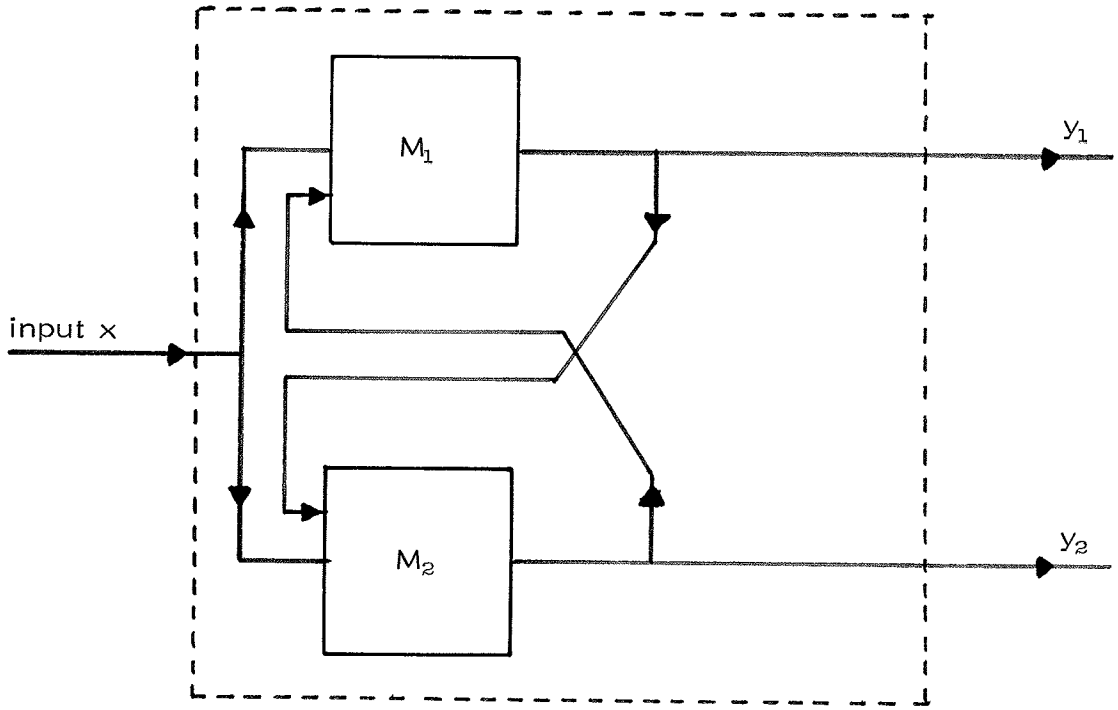


Fig. 6

A graphical illustration of a maximal interconnection of two SFS's M_1 and M_2.

For this kind of decomposition Paz [9] has shown the following:

Theorem 7.2.

Let $M = (S, X, \{M(x)\})$ be an n-state SFS. Then there exists a 2-state SFS A, an (n-1)-state SFS B and a partition ω on $S_A \times S_B$

such that M is state-equivalent to the maximal interconnection of A and B with output-partition ω .

proof Let $S = \{s_1, s_2, \ldots, s_n\}$. s_n is now split according to the previous section into n-1 states, $t_1, t_2, \ldots, t_{n-1}$, resulting in a 2(n-1)-state system M' . The idea in the proof is to construct the transition probabilities in M' in such a way that M will be state-equivalent to M' with output-partition $\omega = \{\overline{s_1}, \overline{s_2}, \ldots, \overline{s_{n-1}}, \overline{t_1, t_2, \ldots, t_{n-1}}\}$, and M' isomorphic to the maximal interconnection of a SFS A with state-set equal to the blocks of $\pi = \{\overline{s_1, s_2, \ldots, s_{n-1}}, \overline{t_1, t_2, \ldots, t_{n-1}}\}$ and B with state-set $\tau = \{\overline{s_1\, t_1}, \overline{s_2\, t_2}, \ldots, \overline{s_{n-1}\, t_{n-1}}\}$. If this is possible the theorem is proved from proposition 2.10.

Let $M(x) = [m_{ij}(x)]$ and $M'(x) = [m'_{ij}(x)]$. Since M' is obtained from M by splitting the state s_n , the following must hold:

$$1) \qquad m'_{ij}(x) = \begin{cases} m_{ij}(x) & \text{for } 1 \le i, j \le n-1 \\ m_{nj}(x) & \text{for } n \le i \le 2(n-1),\ 1 \le j \le n-1 \end{cases}$$

and

$$2) \qquad \sum_{n \le j \le 2(n-1)} m'_{ij}(x) = \begin{cases} m_{in}(x) & \text{for } 1 \le i \le n-1 \\ m_{nn}(x) & \text{for } n \le i \le 2(n-1) \end{cases}$$

On the other hand 1) and 2) ensures that M is state-equivalent to M' with output-partition ω (M is simply the ω-image of M' where ω has

S. P. and is outputconsistent - see the remark after definition 2. 13). From the equation in definition 7. 1. it is seen that a sufficient (and necessary) condition for M' to be maximal decomposable with respect to π and τ is

$$3) \qquad m'_{ij}(x) = \begin{cases} \left(\sum_{j=1}^{n-1} m'_{ij}(x)\right) \cdot \left(m'_{ij}(x) + m'_{i(j+n-1)}(x)\right) & \text{for } 1 \le j \le n-1 \\ \left(\sum_{j=n}^{2(n-1)} m'_{ij}(x)\right) \cdot \left(m'_{ij}(x) + m'_{i(j-n+1)}(x)\right) & \text{for } n \le j \le 2(n-1) \end{cases}$$

(That this condition is sufficient for M' to be maximal decomposable follows from arguments similar to the ones of the proof of theorem 2. 15. - this will, however, be shown later in the proof.)

From the above it follows that it is sufficient to prove the possibility of assigning the matrices M'(x) in such a way that requirements 1), 2) and 3) are met.

1) specifies any entry of M'(x) with $1 \le j \le n-1$. Combining 3), 2) and 1) gives you the following for any entry with $1 \le i \le n$, $n \le j \le 2(n-1)$

$$(*) \qquad m'_{ij}(x) = \left(\sum_{j=n}^{2(n-1)} m'_{ij}(x)\right) \cdot \left(m'_{ij}(x) + m'_{i(j-n+1)}(x)\right) =$$

$$m_{in}(x) \cdot \left(m'_{ij}(x) + m_{i(j-n+1)}(x)\right)$$

$$\Downarrow$$

$$m'_{ij}(x)\,(1 - m_{in}(x)) = m_{in}(x) \cdot m_{i(j-n+1)}(x)$$

If $m_{in}(x) \neq 1$ this equation specifies $m'_{ij}(x)$ - if $m_{in}(x) = 1$ then $m'_{ij}(x)$ can be set arbitrarily as long as it is a legal probability value and $\sum_{j=n}^{2(n-1)} m'_{ij}(x) = 1$.

Finally $m'_{ij}(x)$ is assigned for $n \leq i, j \leq 2(n-1) : m'_{ij}(x) = m'_{nj}(x)$, where $m'_{nj}(x)$ is specified above. (It follows that the last n-1 rows of $M'(x)$ are equal).

With this assignment of $M'(x)$ [1] obviously holds and for $1 \leq i \leq n$

$$\sum_{j=n}^{2(n-1)} m'_{ij}(x) = \sum_{j=n}^{2(n-1)} \frac{m_{in}(x) \cdot m_{i(j-n+1)}(x)}{1-m_{in}(x)} = \frac{m_{in}(x) \cdot (1-m_{in}(x))}{1-m_{in}(x)} = m_{in}(x)$$

if $m_{in}(x) \neq 1$ and the sum is equal to 1 otherwise, which shows that [2] holds. Reversing the argument (*) above gives you immediately that the second line of [3] holds - that the first line also holds follows from:

$$\left(\sum_{j=1}^{n-1} m'_{ij}(x)\right) \cdot \left(m'_{ij}(x) + m'_{i(j+n-1)}(x)\right) = \left(1 - \sum_{j=n}^{2(n-1)} m'_{ij}(x)\right) \cdot \left(m'_{ij}(x) + m'_{i(j+n-1)}(x)\right) =$$

$$m'_{ij}(x) + m'_{i(j+n-1)}(x) \cdot \left(\sum_{j=n}^{2(n-1)} m'_{ij}(x)\right) \cdot \left(m'_{ij}(x) + m'_{i(j+n-1)}(x)\right) =$$

$$m'_{ij}(x) + m'_{i(j+n-1)}(x) - m'_{i(j+n-1)}(x) = m'_{ij}(x) \ .$$

Now from these assignments systems A and B are easily constructed - the state-sets of A and B being the sets of blocks of π and τ respectively. The transition-matrices of A are determined by

$$a_{ij}(x,k) = \sum_{q \in \pi_j} m'_{pq}(x) \qquad \text{for } p = \pi_i \cap \tau_k$$

and

$$b_{ij}(x,k) = \sum_{q \in \tau_j} m'_{pq}(x) \qquad \text{for } p = \pi_k \cap \tau_i$$

From these definitions the transition-matrices of the maximal interconnection of A and B are seen to equal the transition-matrices of

M' and this completes the proof of theorem 7. 2.

Now, given an arbitrary n-state SFS. Apply theorem 7. 2. to it, next to the (n-1)-state system, and so on until you obtion a maximal interconnection of all 2-state-machines. This process can at most go on n-2 steps, so you have the following as an immediate corollary to theorem 7. 2. :

Theorem 7. 3.

Let M be any n-state SFS with input alphabet X. Then there exist n-1 2-state SFS's, M_1 , M_2 , ... , M_{n-1} , the i'th of which with stateset S_i and input alphabet $X \times S_1 \times \ldots \times S_{i-1} \times S_{i+1} \times \ldots \times S_{n-1}$ and a partition ω on $S_1 \times \ldots \times S_{n-1}$ such that M is state-equivalent to the maximal interconnection of the M_i's with ω as output-partition.

Example 8

Let M be a 4-state, one-input (a) SFS with transition-matrix

$$M(a) = \begin{bmatrix} 1/4 & 1/4 & 0 & 1/2 \\ 1/3 & 1/6 & 1/6 & 1/3 \\ 0 & 0 & 0 & 1 \\ 0 & 1/2 & 1/4 & 1/4 \end{bmatrix}$$

Applying the algorithm of theorem 7. 2. to this system gives you the following M'-system - obtained by splitting the fourth state of M

into three states

$$M'(a) = \begin{bmatrix} 1/4 & 1/4 & 0 & 1/4 & 1/4 & 0 \\ 1/3 & 1/6 & 1/6 & 1/6 & 1/12 & 1/12 \\ 0 & 0 & 0 & \underline{0} & \underline{1} & \underline{0} \\ 0 & 1/2 & 1/4 & 0 & 1/6 & 1/12 \\ 0 & 1/2 & 1/4 & 0 & 1/6 & 1/12 \\ 0 & 1/2 & 1/4 & 0 & 1/6 & 1/12 \end{bmatrix}$$

where the underlined entries are don't care entries. M is state-equivalent to this system with output partition $\omega = \{\overline{1}, \overline{2}, \overline{3}, \overline{456}\}$, and M' is isomorphic (and thus state equivalent) to the maximal interconnection of the following 2-state (corresponding to $\pi = \{\overline{123}, \overline{456}\}$) SFS A and 3-state ($\tau = \{\overline{14}, \overline{25}, \overline{36}\}$) SFS B (also constructed from the algorithm of theorem 7. 2.):

Machine A:

$$A(a, 1) = \begin{bmatrix} 1/2 & 1/2 \\ 3/4 & 1/4 \end{bmatrix} \quad A(a, 2) = \begin{bmatrix} 2/3 & 1/3 \\ 3/4 & 1/4 \end{bmatrix} \quad A(a, 3) = \begin{bmatrix} 0 & 1 \\ 3/4 & 1/4 \end{bmatrix}$$

(the 1, 2 and 3 refer to the states of B - $\overline{14}$, $\overline{25}$ and $\overline{36}$ respectively)

Machine B:

$$B(a, 1) = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 1/4 & 1/4 \\ 0 & 1 & 0 \end{bmatrix} \qquad B(a, 2) = \begin{bmatrix} 0 & 2/3 & 1/3 \\ 0 & 2/3 & 1/3 \\ 0 & 2/3 & 1/3 \end{bmatrix}$$

(here the 1 and 2 refer to the states of A - $\overline{123}$ and $\overline{456}$ respectively)

Now, if the same process is used on machine B above, the result will be a maximal interconnection of 3 two-state SFS's to which M is state equivalent.

Now, to continue the line of this paper it would be interesting to see what happens for an interconnection type, in which each component does not only know about the present state of all the others (as in a maximal interconnection), but in which a "flow in the runs of the components" is introduced, such that when the i'th component is to make its transition on an input it knows also about how the j'th component reacted on this input for every $j < i$, i.e. it knows the present and the next state of all j'th components for $j < i$ and the present state of all j'th components for $j \geq i$.

Definition 7.4.

Let $A = (S_A, X_A, \{A(x_A)\})$ and $B = (S_B, X_B, \{B(x_B)\})$ be two SFS's with $X_A = X \times S_B$ and $X_B = X \times S_A \times S_A$ where X is an external input alphabet. The super-maximal interconnection of A and B is defined as the SFS $C = (S_A \times S_B, X, \{C(x)\})$ where

$$C(x) = [c_{ik,jl}(x)] = [a_{ij}(x,k) \cdot b_{kl}(x,i,j)]$$

As for definition 7.1. this definition is extendable to define super-maximal interconnection of more than two SFS's. Since the concepts might be a little confusing I shall state this definition even if definition 7.4. then might seem superfluous.

Definition 7.5.

Let $M_1, M_2, \ldots, M_n$ be a set of SFS's, $M_i = (S_i, X_i, \{M^i(x_i)\})$ where $X_i = X \times S_1 \times \ldots \times S_{i-1} \times S_{i+1} \times \ldots \times S_n \times S_1 \times \ldots \times S_{i-1}$. The super-maximal interconnection of the M_i's is defined as the SFS $M = (S_1 \times \ldots \times S_n, X, \{M(x)\})$ where

$$M(x) = [m_{s_1 s_2 \ldots s_n, s'_1 s'_2 \ldots s'_n}(x)] =$$

$$[m^1_{s_1 s'_1}(x, s_2, s_3, \ldots, s_n) \cdot m^2_{s_2 s'_2}(x, s_1, s_3, \ldots, s_n, s'_1) \cdot \ldots$$

$$\ldots \cdot m^n_{s_n s'_n}(s, s_1, \ldots, s_{n-1}, s'_1, \ldots, s'_{n-1})]$$

Interpretation

The first occurence of S_j in X_i $(j < i)$ is supposed to represent the present state of M_j - the second occurence the next state. The super-maximal interconnection of n SFS's is thus seen to require n time-intervals to make all it's transitions.

When Gelenbe introduced the carry of the next-state in the cascade-decomposition scheme he found that what was gained was a smaller set of conditions for the decomposition to be possible. What happens in this case? Do you gain anything by introducing this next-state flow for the maximal-interconnection scheme, and if you do, what do you gain?

It turns out that you can obtain a reduction in the number of 2-state machines necessary to realize a given machine. This is made clear in the following theorem:

Theorem 7.6.

Let M be any n-state SFS with input-alphabet X . Then there exists a set of N 2-state SFS's $\{M_i\}_{i=1}^{N}$, $M_i = (S_i, X_i, \{M^i(x_i)\})$ where $X_i = X \times S_1 \times \ldots \times S_{i-1} \times S_{i+1} \times S_n \times S_1 \times \ldots \times S_{i-1}$ such that $^{1)}N \leq \log_2(n-1) + 1$ and $^{2)}M$ is isomorphic to a persistent subsystem of the super-maximal interconnection of the M_i's .

Notice that from proposition 2.10. theorems 7.3. and 7.6. are directly comparable. Theorem 7.6. is a consequence of the following theorem 7.7.

Notation

Let n be any positive integer. Then $[n]$ denotes the least integer greater than or equal to $n/2$, i.e.

$$[n] = \begin{cases} n/2 & \text{if } n \text{ is even} \\ (n+1)/2 & \text{if } n \text{ is odd.} \end{cases}$$

Theorem 7.7.

Let M be an n-state SFS with input-alphabet X . Then there exist a 2-state SFS $A = (S_A, X_A, \{A(x_A)\})$ with $X_A = X \times S_B$ and an $[n]$-state SFS $B=(S_B, X_B, \{B(x_B)\})$ with $X_B = X \times S_A \times S_A$, such that M is isomorphic to a persistent subsystem of the super-maximal interconnection of A and B.

First I shall give a proof of theorem 7.7. - probably the reader will be familiar with the arguments necessary to see that theorem 7.6. is implied directly - anyway I shall shortly sketch the arguments afterwards.

proof of theorem 7.7. If n is odd, add one single state to M - obtaining an n+1 - state system M' where

$$\forall x \in X \quad \forall 1 \leq i \leq n : m'_{i(n+1)}(x) = 0$$

and where the transition probabilities from state n+1 are randomly set. If n is even then set system M' = M . In either case it follows that M is isomorphic to a persistent subsystem of M' and that $[n'] = [n] = n'/2$, where n' is the number of states in M' . Let $M' = (S', X, \{M'(x)\})$ with $S' = \{s_1, s_2, \ldots, s_{n'}\}$. Define two partitions
$\pi = \{\overline{s_1 s_2 \ldots s_{[n']}}, \overline{s_{[n']+1} \ldots s_{n'}}\} = \{\pi_1, \pi_2\}$ and
$\tau = \{\overline{s_1 s_{[n']+1}}, \ldots, \overline{s_{[n']} s_n}\} = \{\tau_1, \ldots, \tau_{[n']}\}$ and set the state set of A to the blocks of π - the state-set of B to the blocks of τ . Define the transition probabilities of A and B as follows:

$$a_{ij}(x,k) = \sum_{q \in \pi_j} m'_{pq}(x) \qquad \text{for } p = \pi_i \cap \tau_k$$

$$b_{kl}(x,i,j) = \begin{cases} \dfrac{m'_{pr}(x)}{\sum_{q \in \pi_j} m'_{pq}(x)} & \text{for } p = \pi_i \cap \tau_k,\ r = \pi_j \cap \tau_l \\ & \text{if } \sum_{q \in \pi_j} m'_{pq}(x) \neq 0 \\ \text{don't care} & \text{otherwise} \end{cases}$$

With this assignment I have for each element in the super-maximal interconnection of A and B:

$$c_{ik,jl}(x) = a_{ij}(x,k) \cdot b_{kl}(x,i,j) =$$

$$\sum_{q \in \pi_j} m'_{pq}(x) \cdot \frac{m'_{pr}(x)}{\sum_{q \in \pi_j} m'_{pq}(x)} = m'_{pr}(x) \quad \text{for} \quad p = \pi_i \cap \tau_k \,, \; r = \pi_j \cap \tau_l$$

if $b_{kl}(x,i,j)$ is not a don't-care entry and $c_{ik,jl}(x) = 0$ otherwise. This implies directly that M' is isomorphic to the super-maximal interconnection of A and B with isomorhism $\varphi : S' \to \pi \times \tau$, $\varphi(s) = (\pi_i, \tau_j)$ where $s = \pi_i \cap \tau_j$.

Proposition 2.10 gives you from the above that M is isomorphic to a persistent subsystem of the super-maximal interconnection of A and B - and this completes the proof.

proof of (theorem 7.7. ⇒ theorem 7.6.) Obviously the idea is to apply theorem 7.7. to the given SFS, then to the [n]-state tail-machine and so on. The only thing to prove is that this process will result in no more than $\log_2(n-1) + 1$ components. Let's turn the question for the process and ask for any i the minimal number of states of an original SFS that will result in at least i 2-state components. Denote this number g(i). It is now seen from the description of one step of the process in the proof of theorem 7.7. that g(i) can be expressed in the following recurrence-equation:

$$g(i+1) = 2 \cdot g(i) - 1$$

with initial condition $g(1) = 2$. This equation has the solution $g(i) = 2^{i-1} + 1$, which means that if for any n,N is the number of 2-state components produced by the super-maximal decomposition algorithm then

$$n \geq 2^{N-1} + 1$$
$$\Downarrow$$
$$N \leq \log_2 (n-1) + 1$$

Example 9

Let me take the 4-state SFS of example 8 and decompose it in one step into a super-maximal interconnection of 2 2-state SFS's . 4 is an even integer so the construction of the two components A and B starts directly from the transition-matrix of M :

$$M(a) = \begin{bmatrix} 1/4 & 1/4 & 0 & 1/2 \\ 1/3 & 1/6 & 1/6 & 1/3 \\ 0 & 0 & 0 & 1 \\ 0 & 1/2 & 1/4 & 1/4 \end{bmatrix}$$

The partitions π and τ equal $\{\overline{12}, \overline{34}\}$ and $\{\overline{13}, \overline{24}\}$, respectively.

Front machine A

has two states ($\overline{12}$ and $\overline{34}$), two input-symbols ($\{a\} \times \{\overline{13}, \overline{24}\} = \{a\} \times \{1, 2\}$) and transition-matrices:

$$A(a, 1) = \begin{bmatrix} 1/2 & 1/2 \\ 0 & 1 \end{bmatrix} \qquad A(a, 2) = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

Tail machine B

has two states ($\overline{13}$ and $\overline{24}$), 4 input-symbols ($\{a\} \times \{\overline{12}, \overline{34}\} \times \{\overline{12}, \overline{34}\} = \{a\} \times \{1, 2\} \times \{1, 2\}$) and transition-matrices:

$$B(a, 1, 1) = \begin{bmatrix} 1/2 & 1/2 \\ 2/3 & 1/3 \end{bmatrix} \qquad B(a, 1, 2) = \begin{bmatrix} 0 & 1 \\ 1/3 & 2/3 \end{bmatrix}$$

$$B(a, 2, 1) = \begin{bmatrix} \underline{1} & \underline{0} \\ 0 & 1 \end{bmatrix} \qquad B(a, 2, 2) = \begin{bmatrix} 0 & 1 \\ 1/2 & 1/2 \end{bmatrix}$$

References:

1. Bacon, G.C.: The decomposition of stochastic automata. Information and Control, 7, 1964, p. 320-340.

2. ElGhoroury, H.N. and Gupta, S.C.: Realization of stochastic automata. IEEE Transactions on Computers, vol. C-20, no. 8, 1971, p. 889-893.

3. Gelenbe, E.S.: On the loop-free decomposition of stochastic finite-state systems. Information and Control, 17, 1970, p. 474-485.

4. Hartmanis, J. and Stearns, R.E.: Algebraic structure theory of sequential machines. Prentice Hall, 1966.

5. Heller, A.: Probabilistic automata and stochastic transformations. Math. Syst. Theory, 1, 1967, p. 197-209.

6. Kemeny and Snell: Finite Markov Chains. D. von Nostrands Company, 1960.

7. Kuich, W. and Walk, K.: Block-stochastic matrices and associated finite-state languages. Computing, 1, 1966, p. 50-62.

8. Paz, A.: Introduction to probabilistic automata. Academic Press, 1971.

9. Paz, A.: Whirl decomposition of stochastic automata. Tech. Rep. no 8, Technion Dept. of Comp. Sci.

10. Pugachev, V.S.: Stochastic systems and their connections. Problems of Cont. and Inf. Theory, 1, 1972, p. 55-77.

11. Salomaa, A.: On events represented by probabilistic automata of different types. Canad. Journal of Math., 20, 1966, p. 242-251.

12. Santos, E.S.: Algebraic structure theory of stochastic machines. Math. Syst. Theory, vol 6, no 3, 1972, p. 243-263.