

MAURER'S THEORY OF INSTRUCTIONS

In a recent article, W. D. Maurer (1966) has proposed a mathematical description of computers relating the notions of memory and instructions in a manner which, at least formally, is rather different from the approach taken here. However, as the motivation behind his formulation is similar to ours in some respects, there are interesting correspondences, and as we shall see, several of Maurer's definitions are special cases of those given in this dissertation.

According to Maurer's definition, a computer (which we shall henceforth call an M-computer to avoid confusion with the definition in Chapter IV) is specified by a 4-tuple $(M, \mathcal{B}, \mathcal{S}, \mathcal{I})$ where

- (i) M is a set called the memory,
- (ii) \mathcal{B} is a collection of sets $\{B_x\}$ called base-sets indexed over M ,
- (iii) \mathcal{S} is a subset of $X\mathcal{B}$: that is, \mathcal{S} is a collection of mappings $Q: M \rightarrow \bigcup_{x \in M} B_x$ such that $Q(x) \in B_x$ for all x . The elements of \mathcal{S} are called states.
- (iv) \mathcal{I} is a collection of (total) functions $I: \mathcal{S} \rightarrow \mathcal{S}$ called instructions.

An immediate difference between a computer as previously defined and an M-computer is that an M-computer does not include the specification of a global transition function. In order to determine the next

state of an M-computer both the current state and the instruction being executed must be known, whereas in our definition the latter was completely determined by the former. Thus in order to describe a computer as an M-computer we must modify either the notion of state or instruction. We could, for example, take the mappings $I: \mathcal{S} \rightarrow \mathcal{S}$ to be partial functions (although then Maurer's theorems on the composition of instructions will not hold). In this case, for any computer $\mathcal{C} = (S, f, \Delta, \omega)$ we can by relabelling obtain an M-computer whose states \mathcal{S} correspond one-to-one to the states of \mathcal{C} , whose memory is the set of registers of \mathcal{C} , such that for each instruction Z of \mathcal{C} there is a (partial) mapping $I_Z: \mathcal{S} \rightarrow \mathcal{S}$ which coincides with f on Z .

An alternative approach is to consider the memory of a computer to be just those registers which are not part of any control location, i.e. those in $\overline{\lambda(\omega)}$. Then if Δ is cartesian and the control location is constant for each operation, we can construct an M-computer whose states correspond one-to-one to the blocks of $\overline{\lambda(\omega)}$, such that the mappings I_Z which correspond to the instructions Z of \mathcal{C} are total functions on \mathcal{S} to \mathcal{S} . The formal details are spelled out in A.1.

Maurer further requires that an M-computer satisfy the following two requirements:

(P1) For all $Q_1, Q_2 \in \mathcal{S}$ and $M_1 \subseteq M$, there exists $Q_3 \in \mathcal{S}$ such

that

$$Q_3(x) = Q_1(x) \text{ for } x \in M_1,$$

$$Q_3(x) = Q_2(x) \text{ for } x \in M - M_1;$$

(P2) For all $Q_1, Q_2 \in \mathcal{S}$, $\{x \in M \mid Q_1(x) \neq Q_2(x)\}$ is finite.

(This latter condition clearly corresponds to the finite marking property defined in 3.2.)

The first restriction is particularly relevant to the question of when the state-set \mathcal{S} of an M-computer has a decomposition whose registers correspond to the elements of the memory M. We can, of course, always construct a set of partitions of \mathcal{S} corresponding to M by defining for each $x \in M$ an equivalence relation $\hat{\alpha}_x$ on \mathcal{S} such that

$$Q_1 \hat{\alpha}_x Q_2 \text{ iff } Q_1(x) = Q_2(x), Q_1, Q_2 \in \mathcal{S},$$

and letting α_x be the corresponding partition of \mathcal{S} . Note that if $Q_2 \in \alpha_x Q_1$ for all x , then Q_1 and Q_2 are the same function ($Q_1(x) = Q_2(x)$ for all x) and so $\sum_{x \in M} \alpha_x = \underline{I}$. But in the absence of some further restriction on the state mappings, the α_x may not even be disjoint so we can not in general construct a decomposition with the α_x as atoms.

However, (P1) supplies sufficient structure for the construction to be carried out. Since the α_x are in 1-1 correspondence with the elements of M, we can drop the distinction between M and $\{\alpha_x\}$, and (P1) can be given an equivalent reformulation as a condition on a set of partitions M of a state-set S:

$$(P1)^* \text{ If } M_1 \subseteq M, \left(\bigcap_{a \in M_1} a_s \right) \cap \left(\bigcap_{a \in M - M_1} a_t \right) \text{ is}$$

non-empty for all $s, t \in S$.

By A.3, if M satisfies (P1)*, then M generates a Boolean decomposition Δ of S, and since by A.2, any finite subset of M is permutable, the resulting decomposition is cartesian if M is finite. This, together with the finite marking property (P2), allows us to reverse the construction in A.1(2) and define for any M-computer $(M, \mathcal{R}, \mathcal{S}, \mathcal{C})$ a class of computers which realize it (in the sense that applying the construction of A.1(2) to any member of this class yields the given M-computer).

Briefly, the idea is to add an extra register ω to the set of partitions

$\{\alpha_x\}_{x \in M}$, whose blocks Z_I correspond to the instructions $I \in \mathcal{A}$,

and to define a transition function f on an enlarged set of states $\mathcal{S} \times \mathcal{A}$ such that when the computer state is in the set Z_I , f maps the contents of the memory registers in M in the same way that I does. The details are spelled out in A.4.

Maurer defines for each instruction I of an M -computer two regions or subsets of the memory M . The output region is defined as

$$OR(I) = \{x \in M \mid \text{for some } Q, Q(x) \neq I(Q)(x)\}.$$

The input region is defined as

$$IR(I) = \{x \in M \mid \text{for some } Q_1, Q_2, \text{ there exists } y \in OR(I) \\ \text{such that } Q_1(z) = Q_2(z) \text{ if } z \neq x, \text{ and} \\ I(Q_1)(y) \neq I(Q_2)(y)\}.$$

Clearly, the output region corresponds to the notion of range, and indeed, if we consider a computer of the sort defined in A.4 which realizes a given M -computer, then

$$OR(I) = \phi(\rho(Z_I)) \cap M,$$

where M is the set of memory registers excluding the control location $\lambda(\omega)$.

The concept of input region is related, as one might expect, to the notion of domain. In order to discuss the connection, it is helpful to introduce some further terminology: two states s_1, s_2 will be said to be an α -minimal pair if

$$\alpha s_1 \neq \alpha s_2 \text{ and } \bar{\alpha} s_1 = \bar{\alpha} s_2,$$

and s_1, s_2 will be said to contrast on β if

$$\beta f(s_1) \neq \beta f(s_2).$$

Then the definition of the input region of a computer instruction can be expressed as follows:

a register is in the input region $IR(Z)$
if there exists an α -minimal pair in Z
which contrasts on $\rho(Z)$.

(For the sake of simplicity, we have replaced $OR(Z)$ in Maurer's definition by $\rho(Z)$.)

From A.5, we have that

$$\Sigma IR(Z) \leq \delta^*(Z),$$

(where as before, $\delta^*(Z) = \delta^*(Z; \rho(Z))$), and by A.6, if $(P1)^*$ holds,

$$\Sigma IR(Z) = \delta^*(Z).$$

The advantage of the notion of "input region" as compared with the definition of domain is that it provides a criterion for testing each register α separately to see if knowledge of the contents of α is necessary to predict the next state of $\rho(Z)$. Its disadvantage is that unless the states of the registers are independently assignable, i.e. Δ is cartesian, there may be no α -minimal pairs in Z and so the test cannot be applied.

Note that it does not follow from Maurer's definitions of $IR(I)$ and $OR(I)$, that if for states Q_1 and Q_2 , $IR(I)$ has the same contents, then $OR(I)$ has the same contents for the states $I(Q_1)$ and $I(Q_2)$. As Maurer shows, this property holds only if $(P2)$ holds. This result is in a sense a special case of Theorem 3.3, for by definition, the domain of

an instruction (which includes the input region) has the desired property if it exists, and its existence is assured by the finite-marking property which is equivalent to (P2).

A.1 (1) In order to interpret a computer $\mathcal{C} = (S, f, \Delta, \omega)$ as an M-computer $(M, \mathcal{B}, \mathcal{S}, \mathcal{C})$, let the memory M be the set of registers $A(\Delta)$, and let the base-sets $B_\alpha = \alpha$ for each $\alpha \in A(\Delta)$, so that $M = A(\Delta) = M$.

For each $s \in S$ define

$$s^{\mathcal{H}}: A(\Delta) \rightarrow \bigcup_{\alpha \in A(\Delta)} \alpha \text{ by } s^{\mathcal{H}}(\alpha) = \alpha s$$

and let the set of states \mathcal{S} of the M-computer = $\{s^{\mathcal{H}} | s \in S\}$.

Finally, define the partial functions $I_Z: \mathcal{S} \rightarrow \mathcal{S}$ such that

$$I_Z(s^{\mathcal{H}}) = t^{\mathcal{H}} \text{ where } t^{\mathcal{H}}(\alpha) = f_{Z; \alpha}(\delta^*(Z; \alpha)s)$$

for all $s \in Z$. Then by the definition of $f_{Z; \alpha}$, when $s \in Z$,

$$[I_Z(s^{\mathcal{H}})](\alpha) = t^{\mathcal{H}}(\alpha) = \alpha f(s).$$

(2) An alternative interpretation is obtained by setting

$M = \phi(\overline{\lambda(\omega)})$ and $\mathcal{B} = M$. For $X = \overline{\lambda(\omega)}s$, define $X^{\mathcal{H}}(\alpha) = \alpha s$ for each register α in $\phi(\overline{\lambda(\omega)})$, and let the set of states \mathcal{S} of the M-computer = $\{X^{\mathcal{H}} | X \in \overline{\lambda(\omega)}\}$. If Δ is cartesian and $\lambda(\omega) = \lambda(A)$ for all $A \in \omega$, then $\delta^*(Z; \alpha) \subseteq \overline{\lambda(\omega)}$ for each instruction Z, and for each $Y \in \delta^*(Z; \alpha)$, $Y \cap Z$ is non-empty.

Hence $f_{Z; \alpha}$ is defined on all blocks of $\delta^*(Z; \alpha)$. Define

$$I_Z: \mathcal{S} \rightarrow \mathcal{S} \text{ by setting } [I_Z(X^{\mathcal{H}})](\alpha) = f_{Z; \alpha}(Y) \text{ whenever } X \subseteq Y.$$

Since $\delta^*(Z; \alpha) \subseteq \overline{\lambda(\omega)}$, every block $X \in \overline{\lambda(\omega)}$ is contained in exactly one block of $\delta^*(Z, \alpha)$ and so I_Z is well-defined for all $X^{\mathcal{H}} \in \mathcal{S}$, and as before, if $s \in Z$ and $X = \overline{\lambda(\omega)}s$, then

$$[I_Z(X^{\mathcal{H}})](\alpha) = \alpha f(s), \text{ for each } \alpha \in M.$$

A.2 Theorem: If M is a collection of partitions of S such that (P1)* holds, then any finite subset of M is permutable.

Proof: Let $F = [\alpha_1, \dots, \alpha_n]$ and let $X_i \in \alpha_i$. Pick $s_0 \in X_1$, and $s_1 \in X_2$. By hypothesis, there exists s_2 such that $\alpha_1 s_2 = \alpha_1 s_0 = X_1$ and $\alpha s_2 = \alpha s_1$ for $\alpha \neq \alpha_1$. In particular, $\alpha_2 s_2 = \alpha_2 s_1 = X_2$, so $X_1 \cap X_2$ is non-empty. Repeating this argument a finite number of times, we have that if $s_{n-2} \in \bigcap_{i < n} X_i$ and $s_{n-1} \in X_n$, there exists s_n such that

$$\alpha_i s_n = \alpha_i s_{n-2} = X_i \text{ for } i < n, \text{ and}$$

$$\alpha_n s_n = \alpha_n s_{n-1} = X_n,$$

so $\bigcap_{i \leq n} X_i$ is non-empty.

A.3 Theorem: If M is a collection of partitions of S such that (P1)* holds and $\Sigma M = \underline{I}$, then there exists a Boolean decomposition Δ of S such that $A(\Delta) = M$.

Proof: If M is finite, then by A.2, (P1)* insures that M is a permutable set and so Theorem 2.17 gives the desired result. In this case, Δ is cartesian.

If M is infinite, we can still use the proof pattern of 2.17 by defining $\Delta = \{\underline{0}\} \cup \{\Sigma N \mid \emptyset \neq N \subseteq M\}$, and showing that Δ is lattice-isomorphic to $B(M)$. However, as now arbitrary subsets of M are not permutable, the proof of 2.17 must be modified in two places, using (P1)*:

In order to show that M is an independent set, we need to show (see proof of 2.14)

$$\alpha' s_1 \cap \left(\bigcap_{\alpha \neq \alpha'} \alpha s_2 \right) \neq \emptyset \text{ for all } s_1, s_2 \in S, \alpha, \alpha' \in M.$$

But this follows immediately from (P1)*. Likewise, the proof that Δ is closed under meets rests on the existence of $r \in \left(\bigcap_{\alpha \in X} \alpha s \right) \cap \left(\bigcap_{\alpha \in Y} \alpha t \right)$ where X and Y are disjoint subsets of M , and this is guaranteed by (P1)*.

A.4 In order to define a computer which realizes a given M -computer $(M, \mathcal{S}, \mathcal{I}, \mathcal{d})$;

(i) let $S = \mathcal{S} \times \mathcal{d}$;

(ii) let f be any function on S to S satisfying

$$f(Q, I_1) = (I_1(Q), I_2) \text{ for all } Q \in \mathcal{S}, I_1 \in \mathcal{d};$$

(iii) define for each $x \in M$, $\hat{\alpha}_x$ to be the equivalence relation $(Q_1, I_1) \hat{\alpha}_x (Q_2, I_2)$ iff $Q_1(x) = Q_2(x)$;

(iv) let $Z_I = \{s \in S \mid s = (Q, I)\}$ and define $\omega = \{Z_I\}_{I \in \mathcal{d}}$.

Then since M satisfies (P1), it can be easily seen that $M^* = \left(\{\alpha_x\}_{x \in M} \right) \cup \{\omega\}$ satisfies (P1)* and hence generates a Boolean decomposition Δ with M^* as the set of registers. As the M -computer satisfies (P2), Δ has the finite marking property, and so (S, f, Δ, ω) is a computer. In this computer, the control location $\lambda(\omega) = \omega$, and so its instructions are the blocks Z_I .

Furthermore, the following identities hold:

$$\phi(\overline{\lambda(\omega)}) = \phi(\overline{\omega}) = \{\alpha_x\}_{x \in M}, \text{ and if}$$

$X \in \overline{\lambda(\omega)}s$ and $s = (Q, I)$ then $X^{\#} = Q$, where X is defined as in A.1(2). As a further consequence of (P1)* we have that ω and $\overline{\omega}$ are permutable, and ω and $\delta^*(Z_I; \alpha_x) \leq \overline{\omega}$ are permutable. Hence $f_{Z_I; \alpha_x}(Y)$ is defined for all $Y \in \delta^*(Z_I; \alpha_x)$. Thus, for all $s = (Q, I) \in Z_I$, $f_{Z_I; \alpha_x}(Y) = \alpha_x f(s) = \alpha_x(I(Q), J)$ and this by definition is the block of α_x which corresponds to $I(Q)(x)$.

A.5 Theorem: $\Sigma IR(Z) \leq \delta^*(Z)$.

Proof: If a register α is not in $\phi(\delta^*(Z))$, and s_1 and s_2 are an α -minimal pair in Z , then

$$\delta^*(Z)s_1 = \delta^*(Z)s_2$$

and so $\rho(Z)s_1 = \rho(Z)s_2$. Hence s_1 and s_2 do not contrast on $\rho(Z)$.

So $\alpha \notin \phi(\delta^*(Z))$ implies $\alpha \notin IR(Z)$ and $IR(Z) \subseteq \phi(\delta^*(Z))$,

or equivalently $\Sigma IR(Z) \leq \Sigma \phi(\delta^*(Z)) = \delta^*(Z)$.

A.6 Theorem: If $A(\Delta)$ satisfies (P1)*, $\Sigma IR(Z) = \delta^*(Z)$.

Proof: By A.5, it suffices to show $\phi(\delta^*(Z)) \subseteq IR(Z)$. Pick $\alpha \in \phi(\delta^*(Z))$, and let X, Y be distinct blocks of α . By hypothesis, any finite collection of disjoint components in Δ is permutable, and so, as $Z \in \lambda(Z)$, and $\lambda(Z) \cdot \alpha = \underline{0}$, there exist

$$s_1 \in Z \cap X \cap \overline{(\lambda(Z) \cdot \alpha)}s,$$

$$s_2 \in Z \cap Y \cap \overline{(\lambda(Z) \cdot \alpha)}s,$$

for any s . By construction s_1 and s_2 are an α -minimal pair in Z . Suppose every such α -minimal pair fail to contrast on $\rho(Z)$. Then for all $s \in Z$, $t \in \overline{\alpha}s$ implies $\rho(Z)s = \rho(Z)t$ and hence $\overline{\alpha} \xrightarrow{P} \rho(Z)$

on Z . Since $\alpha \in \phi(\delta^*Z)$, $\bar{\alpha} \geq \lambda(Z)$ and therefore $\bar{\alpha} \geq \delta(Z) \geq \delta^*(Z)$.
But $\alpha \cdot \bar{\alpha} = \underline{0}$, so $\alpha \cdot \delta^*(Z) = \underline{0}$, which is a contradiction. Hence
at least one α -minimal pair in Z contrasts on $\rho(Z)$ and
 $\alpha \in \text{IR}(Z)$.

RANGES AND DOMAINS OF TYPICAL
COMPUTER INSTRUCTIONS

Z	$\rho(Z)$	$\delta(Z)$	$\delta^*(Z;AC)$	$\delta^*(Z;100)$	$\delta^*(Z;R)$	$\delta^*(Z;L)$
SSP	R+L+AC	R+L+M	<u>0</u>	100	L+M	L
STZ 100	R+L+100	R+L+M	AC	<u>0</u>	L+M	L
STO 100	R+L+100	R+L+M+AC	AC	AC	L+M	L
STO* 100	R+L+M	R+L+M+AC+XRS	AC	M+AC+XRS	L+M	L
ADD 100	R+L+AC	R+L+M	100	100	L+M	L
ADD 100,1	R+L+AC	R+L+M+XR1	M+XR1	100	L+M	L
ADD* 100	R+L+AC	R+L+M+XRS	M+XRS	100	L+M	L
XEC 100	R	R+100	AC	100	100	L
TRA 100	R+L	R+100	AC	100	100	<u>0</u>
TRA* 100	R+L	R+M+XRS	AC	100	M+XRS	M+XRS
TZE 100	R+L	R+M+L+AC	AC	100	M+L+AC	M+L+AC

The above instructions are from the repertoire of the IBM 709/90 and are written in the usual assembly-language format.

R = storage register containing the instruction currently being executed = $\lambda(Z)$;

L = location counter; XR1 = index register 1, XRS = index registers 1,2,4;

AC = accumulator; M = addressable core storage = registers $0-7777_8$.

The operation codes have the following interpretations:

SSP = set sign of AC to plus;

STZ 100 = store zero in register 100;

STO 100 = store contents of AC in 100;

ADD 100 = add contents of 100 to AC;

XEC 100 = execute instruction in 100;

TRA 100 = transfer to register 100;

TZE 100 = transfer to 100 if AC contains zero.

THE INFORMATION MEASURE H APPLIED TO PARTITIONS

If S is finite, we define

$$H(\alpha) = - \sum_{X \in \alpha} \frac{|X|}{|S|} \log \frac{|X|}{|S|}$$

for $\alpha \in P(S)$, (where $|X|$ = number of elements in X). The connection between the information measure H and the partial ordering of partitions of S is summarized in the following theorem:

- (a) $\alpha \leq \beta$ implies $H(\alpha) \leq H(\beta)$,
- (b) $H(\alpha + \beta) \leq H(\alpha) + H(\beta) - H(\alpha.\beta)$,
- (c) $H(\alpha + \beta) = H(\alpha) + H(\beta) - H(\alpha.\beta)$ for all $\alpha, \beta \in \Delta$,

iff Δ is cartesian.

Proof: (a) Let X be a block of α , and let X_1, \dots, X_K be the blocks of β contained in X . Setting $h(u) = -u \log u$, it suffices to show

$$\sum_1^K h(|X_i|/|S|) \geq h(|X|/|S|)$$

for each $X \in \beta$. But $h(u+v) = -u \log(u+v) - v \log(u+v)$
 $\leq -u \log u - v \log v = h(u) + h(v)$,

and so by finite induction

$$\sum_1^K h(|X_i|/|S|) \geq h\left(\sum_1^K |X_i|/|S|\right).$$

Since the X_i are disjoint and their union is X , $\sum_1^K |X_i| = |X|$ and

so

$$\sum_1^K h(|X_i|/|S|) \geq h(|X|/|S|).$$

(b) Let us assign to each $s \in S$ the uniform probability $p(s) = \frac{1}{|S|}$.

Then we can treat a partition α of S as a finite probability scheme in the sense of Khinchin (1957, p. 2), the probability of a block $X \in \alpha$ being $|X|/|S|$. The conditional probabilities of $Y \in \beta$ given X ,

$$p(Y|X) = |X \cap Y|/|X|$$

are then exactly the probabilities associated with the partition $\beta|X$ defined in 5.9. Hence $H(\beta|X)$ agrees with the customary definition of the conditional information in β given the event X . This allows us to define in the usual way

$$H(\alpha|\beta) = \sum_{X \in \beta} \frac{|X|}{|S|} H(\alpha|X).$$

Note that $\beta \geq \alpha$ iff $X \in \beta$, $Y \in \alpha$ implies either $|Y \cap X| = 0$ or $|Y \cap X| = |X|$; and therefore $\beta \geq \alpha$ is equivalent to $H(\alpha|\beta) = 0$.

The probability of the event "X and Y" is $p(X \cap Y)$ and so the joint probabilities of $X \in \alpha$, $Y \in \beta$ correspond to the probabilities associated with the partition $\alpha + \beta$. Thus $H(\alpha + \beta)$ coincides with the usual definition of the joint information in α and β (Khinchin, 1957, p. 5).

As a consequence of the above identities, we can apply a number of known results derived for the information measure defined on probability schemes to the H measure defined on partitions.

In particular, $H(\alpha + \beta) = H(\alpha) + H(\beta) - T(\alpha, \beta)$, where

$$T(\alpha, \beta) = T(\beta, \alpha) = H(\beta) - H(\beta|\alpha).$$

Let us define

$$T_{\gamma}(\alpha, \beta) = H(\alpha|\gamma) - H(\alpha|\beta + \gamma).$$

and

$$I(\alpha, \beta, \gamma) = T_{\gamma}(\alpha, \beta) - T(\alpha, \beta)$$

for arbitrary $\gamma \in P(S)$. Then since $T_{\gamma}(\alpha, \beta)$ can be shown to be always non-negative (Khinchin, 1957, p. 37),

$$I(\alpha, \beta, \gamma) \geq -T(\alpha, \beta).$$

By a result due to McGill (1954), we have also

$$T(\alpha + \beta, \gamma) = T(\alpha, \gamma) + T(\beta, \gamma) + I(\alpha, \beta, \gamma).$$

If $\gamma = \alpha.\beta$, then $\gamma \leq \alpha$, so $H(\gamma|\alpha) = 0$, and

$$T(\alpha, \gamma) = H(\gamma) - H(\gamma|\alpha) = H(\gamma).$$

Likewise, $T(\beta, \gamma) = H(\gamma)$ and hence

$$T(\alpha + \beta, \gamma) = 2H(\gamma) + I(\alpha, \beta, \gamma).$$

But $T(\alpha + \beta, \gamma) = H(\alpha + \beta) + H(\gamma) - H(\alpha + \beta + \gamma)$

$$= H(\gamma), \text{ since } \alpha + \beta + \gamma = \alpha + \beta,$$

and so $I(\alpha, \beta, \gamma) = -H(\gamma) \geq -T(\alpha, \beta)$. Hence

$$H(\alpha + \beta) = H(\alpha) + H(\beta) - T(\alpha, \beta) \leq H(\alpha) + H(\beta) - H(\alpha.\beta).$$

(c) Suppose Δ is cartesian, and $\alpha, \beta \in \Delta$ are disjoint. Then α and β are permutable and by 2.13, there is a one-to-one correspondence between the blocks of $\alpha + \beta$ and the elements of $\alpha \times \beta$. Hence if α has m blocks and β has n blocks, $\alpha + \beta$ has mn blocks. Since Δ is cartesian, all blocks of a component have the same number of elements,

so $X \in \gamma \in \Delta$ implies $|X| = |S|/K$ where K is the number of blocks in γ .

Hence if $X \in \alpha$, and $Y \in \beta$, $X \cap Y \in \alpha + \beta$, and $|X \cap Y| = |S|/mn$.

Then $p(X \cap Y) = 1/mn$

$$= \left(\frac{|S|/m}{|S|} \right) \left(\frac{|S|/n}{|S|} \right) = p(X)p(Y).$$

But this means that the probability schemes corresponding to α and β are independent (in the probabilistic sense) and hence

$$H(\alpha + \beta) = H(\alpha) + H(\beta)$$

(see Khinchin, 1957, p. 5).

Now let α, β be arbitrary components in Δ . Then we can write

$$\begin{aligned} H(\alpha + \beta) &= H(\alpha \bar{\beta}) + H(\alpha \beta) + H(\bar{\alpha} \beta) \\ &= H(\alpha \bar{\beta}) + H(\alpha \beta) + H(\bar{\alpha} \beta) + H(\alpha \beta) - H(\alpha \beta) \\ &= H(\alpha) + H(\beta) - H(\alpha \beta). \end{aligned}$$

Conversely, suppose $H(\alpha + \beta) = H(\alpha) + H(\beta) - H(\alpha \beta)$ for all $\alpha, \beta \in \Delta$.

Pick $\alpha, \beta \in A(\Delta)$. Then $\alpha \beta = \underline{0}$ and so $H(\alpha + \beta) = H(\alpha) + H(\beta)$. But this implies the schemes corresponding to α and β are independent in the probabilistic sense and therefore

$$p(X|Y) = |X \cap Y|/|Y| = p(X) = |X|/|S| \neq 0,$$

for all $X \in \alpha$, $Y \in \beta$. Thus $X \cap Y$ is non-empty, and α and β are permutable. Since Δ is finite, this implies $A(\Delta)$ is a permutable set, and hence Δ is cartesian.

LIST OF SYMBOLS

<u>Symbol</u>	<u>Meaning</u>	<u>Page</u>
$A(\Delta)$	set of atoms of Δ	17
$C(f)$	maximal weakly connected subsets of S	79
f	transition function	46
$f^n(s)$	state reached in n transitions from s	57
$f_{Z;\beta}$	mapping from $\delta^*(Z;\beta)$ to β	64
\underline{I}	largest element of $P(S)$	15
$K(h)$	kernel of the mapping h	38
L_S	lattice of partitions with S.P.	82
$L(X)$	sub-lattice of L generated by X	18
$\underline{0}$	smallest element of $P(S)$	15
P	stored program	93
$P(S)$	lattice of all partitions of S	13
$\overset{P}{\downarrow}$	prediction relation between partitions	60
S	set of states	10
$s, t, \text{ etc.}$	states in S	
$\overset{T}{\downarrow}$	transmission relation between partitions	65
$X @$	cartesian product of a collection of sets Q	31
$X, Y, Z, \text{ etc.}$	subsets of S	
α, β, γ	partitions in $P(S)$	
αZ	partition obtained by restricting α to Z	70
$\bar{\alpha}$	complement of α	20
Γ	access path	

$\Gamma(s)$	access path whose path-state contains s	94
Δ	decomposition	10, 17
$\lambda(\alpha)$	location of α in Δ	39
$\lambda(X)$	location of $\pi\{X\}$	
$\tilde{\lambda}(\alpha)$	largest component contained in α	79
$v_i(s)$	location of the stored form of the instruction $Z_{i+1}(s)$	98
$\overline{\Pi}_A$	greatest lower bound of the set A	15
$\pi_{\mathcal{Q}}$	partition defined by a collection of disjoint sets \mathcal{Q}	40
πs	block of π containing s	12
πh	mapping of S onto π	12
$\hat{\pi}$	equivalence relation determined by π	12
$\rho(X)$	range of X	60
$\overline{\rho}(X)$	largest component stable on X	59
$\rho_i(s)$	component written in by instructions $Z_1(s), \dots, Z_{i-1}(s)$.	93
ΣA	least upper bound of the set A	15
$\phi(\alpha)$	set of atoms contained in α	18
ϕ_α	filter of components containing α	39
ω	operation repertoire	46
$+$	join	10
\cdot	meet	10
\leq	inclusion relation	10, 13
$\delta(Z)$	$\delta(Z; \rho(Z))$	63
$\delta(Z; \beta)$	β -domain of Z	63
$\delta^*(Z; \beta)$	$\overline{\lambda(Z)} \cdot \delta(Z; \beta)$	64

- Berge, Claude. The Theory of Graphs. Trans. Alison Doig. London: Methuen & Co., Ltd., 1962.
- Birkhoff, Garret. Lattice Theory. 1st ed. revised. (Am. Math. Soc. Coll. XXV.) Providence: American Mathematical Society, 1961.
- Birkhoff, G., and Ward, M. "A Characterization of Boolean Algebras," Ann. Math., XL (1939), pp. 609-610.
- Burks, A.W. "The Logic of Fixed and Growing Automata," in Proceedings of the Symposium on the Theory of Switching. Ed. H. Aiken. Cambridge: 1959, pp. 147-188.
- Clifford, A.H., and Preston, G.B. The Algebraic Theory of Semigroups. Vol. I. (Mathematical Surveys 7.) Providence: American Mathematical Society, 1961.
- Cohn, P.M. Universal Algebra. New York: Harper & Row, 1965.
- Flores, Ivan. Computer Logic. Englewood Cliffs: Prentice-Hall, Inc., 1960.
- George, F.H. The Brain as a Computer. Oxford: Pergamon Press, 1962.
- Hartmanis, J., and Stearns, R.E. "Pair Algebras and Their Application to Automata Theory," Information and Control, VII, No. 4 (Dec. 1964), pp. 485-507.
- _____. Algebraic Structure Theory of Sequential Machines. Englewood Cliffs: Prentice-Hall, Inc., 1966.
- IBM Reference Manual: 709 Data Processing System. International Business Machines, 1961.
- IBM Reference Manual: 1401 Data-Processing System. International Business Machines, 1961.
- IBM 650 Magnetic Drum Data-Processing Machine: Manual of Operation. International Business Machines, 1955.
- Khinchin, A.I. Mathematical Foundations of Information Theory. Trans. R.A. Silverman and M.D. Friedman. New York: Dover Publications, Inc., 1957.

Kleene, S.C. "Representation of Events in Nerve Nets and Finite Automata," in Automata Studies, Ed. C.E. Shannon and J. McCarthy. (Annals of Mathematical Studies 47.) Princeton: Princeton University Press, 1956.

Kochen, Manfred. "Organized Systems with Discrete Information Transfer," General Systems, II, 1957.

Maurer, W.D. "A Theory of Computer Instructions," Journal of the Association for Computing Machinery, XIII, No. 2 (April, 1966), pp. 226-235.

McGill, W.J. "Multivariate Information Transmission," Psychometrika, XIX (1954), pp. 97-116.

Roosen-Runge, P.H. "Toward a Theory of Parts and Wholes: An Algebraic Approach," General Systems, XI (1966), pp. 13-18.

Stearns, R.E., and Hartmanis, J. "On the State Assignment Problem for Sequential Machines II," IRE Trans. on Electronic Computers, Vol. EC-10 (Dec. 1961), pp. 593-603.

Young, J.Z. The Memory System of the Brain. Berkeley: University of California Press, 1966.

Purpose 2
Domain: Range = 6
Complexity 80