MAC TR-105

[illegible] ANALYSIS OF SORTING NETWORKS

Burton J. Smith

October 1972

[illegible] pported in part by the National Science
[illegible] earch grant GJ34671, and in part by the
[illegible] jects Agency of the Department of
[illegible] rder No. 433 which was monitored by ONR
[illegible] 70-A-0362-0001.

[illegible] SETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

[illegible] MASSACHUSETTS 021[illegible]

AN ANALYSIS OF SORTING NETWORKS

ABSTRACT

Comparators which sort two numbers can be interconnected to form networks which sort n numbers for any n. The input and output characteristics of comparator networks are analyzed from several different points of view.

This report reproduces a thesis of the same title submitted to the Department of Electrical Engineering, Massachusetts Institute of Technology, in partial fulfillment of the requirements for the degree of Doctor of Science, August 1972.

ACKNOWLEDGEMENTS

Thanks are due and most gratefully given to Fred Hennie for his insights, help and understanding. I would also like to thank Dave Liu and Peter Elias for the many hours they spent with me on this problem, and Marsha Baker for the many hours she spent making it all legible. Finally, I would like to express my gratitude to my wife Dottie, for bearing it all, and to my daughter Katherine, for comic relief.

TABLE OF CONTENTS

TITLE PAGE 1

ABSTRACT.. 2

ACKNOWLEDGEMENTS.. 3

TABLE OF CONTENTS... 4

CHAPTER 1 - SORTING AND SORTING NETWORKS.................... 5

1.1 Sorting.. 5

1.2 Sorting Networks... 9

1.3 The Analysis Problem for Sorting Networks............... 13

CHAPTER 2 - CRITERIA FOR DECIDING WHETHER A NETWORK SORTS.... 16

2.1 Comparator Networks as Finite State Machines............ 16

2.2 The Zero-one Principle.................................. 25

CHAPTER 3 - COMPARATOR NETWORK OUTPUT BEHAVIOR.............. 35

3.1 Output Characterizations................................ 35

3.2 Zero-one Valued Output Assignments...................... 39

3.3 Injective Output Assignments............................ 51

3.4 The Lattice of Partial Orders........................... 57

CHAPTER 4 - OUTPUTS CHARACTERIZED BY A PARTIAL ORDER......... 71

4.1 Sorting Injective Assignments with respect to a Partial Order.. 71

4.2 Real-valued Assignments and Convexity................... 79

CHAPTER 5 - CONCLUSIONS..................................... 89

BIBLIOGRAPHY.. 92

BIOGRAPHICAL NOTE... 93

CHAPTER 1

SORTING AND SORTING NETWORKS

1.1 Sorting

The ability to sort a sequence of objects seems to be important in the solution of many data processing problems. This thesis deals with a particular kind of sorting algorithm called a sorting network; but before discussing sorting networks, it will be appropriate to deal with sorting *per se*.

Intuitively, to sort is to rearrange a sequence of "values" or "numbers" to conform to some order. There are always two orders involved in this process; the order implicit in the sequence and the numerical order of the values themselves. For example, if each of a sequence of locations in a computer memory contains a number, we can talk about sorting the sequence of numbers. This is done by changing the locations of the numbers so that after the process is completed, the number in location x is less than or equal to the number in location y whenever location x precedes location y. Note that sorting affects neither the sequence of the locations nor the values of the numbers. Rather, it changes the assignment of numbers to locations. Definition 1.1.1 generalizes these intuitive notions to deal with partially ordered sets.

Definition 1.1.1 If D and R are sets partially ordered by P and $\leq$, respectively, and if f is a function from D to R, a permutation π on D *sorts* f (*with respect to P*) if

$$(\forall x, y \in D)(\pi(x) P \pi(y) \Rightarrow f(x) \leq f(y))$$

or, equivalently,

$$(\forall x, y \in D)(xPy \Rightarrow f(\pi^{-1}(x)) \leq f(\pi^{-1}(y)))$$

The function f is called an <u>assignment</u> from the <u>locations</u> of D to the <u>values</u> of R. If f is an assignment which is already sorted with respect to P, i.e. if f satisfies the condition

$$(\forall x, y \in D)(xPy \Rightarrow f(x) \leq f(y))$$

then f is <u>consistent with P</u>. $\square$

Letting $f \cdot g$ denote the composition of the functions f and g, i.e. $f \cdot g(x) = g(f(x))$ for all x in the inverse image of the domain of g, it is easy to see that if π sorts f with respect to P, then $\pi^{-1} \cdot f$ is consistent with P.

<u>Example 1.1.2</u> If D and R are the sets $\{a,b,c,d\}$ and $\{0,1,2,3,4\}$, respectively, with P and $\leq$ the obvious total orders, then the permutation (abd)(c) sorts the assignment f defined by $f(a) = f(d) = 0$, $f(b) = 3$, $f(c) = 2$ with respect to P because $f(\pi^{-1}(a)) = 0$, $f(\pi^{-1}(b)) = 0$, $f(\pi^{-1}(c)) = 2$, and $f(\pi^{-1}(d)) = 3$. $\square$

Of course, the most usual special case of sorting arises when D is a finite set, P is a total order, R is a set of numbers, and $\leq$ is the familiar total order. If D is not finite or if $\leq$ is not total, it may not be possible to sort some kinds of assignments.

Example 1.1.3 Let D and R both be the set of positive integers, with P and $\leq$ the obvious total orderings. The assignment f defined by

$$f(n) = \begin{cases} 0 & \text{if } n \text{ is even} \\ 1 & \text{if } n \text{ is odd} \end{cases}$$

cannot be sorted because $f(n) = 0$ for infinitely many n, and therefore a permutation π to sort f could not have $\pi(n)$ finite for n odd. □

Example 1.1.4 Let D and R both be finite sets, with P any partial order on D and $\leq$ the identity relation on R. Then no injective assignment f can be sorted unless P is the identity relation on D, for if xPy with $x \neq y$, $f(\pi^{-1}(x)) = f(\pi^{-1}(y))$ is false for every permutation π. □

It is clear that if D is finite and P and $\leq$ are a total orders, any assignment from D to R can be sorted by an appropriate permutation. In fact, P need not be a total order, as the next theorem shows.

Theorem 1.1.5 Let f be any assignment from a finite set D to a range R, with P a partial order on D and $\leq$ a total order on R. Then any assignment $f\colon D \rightarrow R$ can be sorted with respect to P.

Proof: Let T be a total order on D such that $P \subseteq T$, and let π sort f with respect to T. Then

$$(\forall x, y \in D)(xTy \Rightarrow f(\pi^{-1}(x)) \leq f(\pi^{-1}(y)))$$

Since $P \subseteq T$, we have

$$(\forall x, y \in D)(xPy \Rightarrow xTy)$$

and

$$(\forall x, y \in D)(xPy \Rightarrow f(\pi^{-1}(x)) \leq f(\pi^{-1}(y)))$$

so π sorts f with respect to P. $\square$

It will be assumed in what follows that D is a finite set and $\leq$ is a total order. It will be useful to let R be some set of numbers, with $\leq$ the familiar total ordering. R will be either a finite set of positive integers, the set of all positive integers, or the set of real numbers; most often, it will be unnecessary to state explicitly which of these sets R denotes. The domain D will usually be a set of numbers, or a set of letters where confusion might result from the use of numbers.

It will also be useful to talk about sets containing all of the sorted assignments of a particular kind.

<u>Definition 1.1.6</u> If P is a partial order on the domain D, then the sets A_P, I_P, and Z_P are defined as follows:

$$A_P = \{f\colon D \to R \mid (\forall x, y \in D)(xPy \Rightarrow f(x) \leq f(y))$$
$$I_P = \{f\colon D \to R \mid f \in A_P \wedge f \text{ is injective}\}$$
$$Z_P = \{f\colon D \to R \mid f \in A_P \wedge \text{Range}(f) = \{0,1\}\}$$

That is, A_P is the set of assignments consistent with P, I_P is the set of injective (one-one) assignments consistent with P, and Z_P is the set of zero-one valued assignments consistent with P. The set of all assignments from D to R will be written $A_=$; similarly, $I_=$ denotes the set of all injective assignments from D to R, and $Z_=$ will denote the set of all zero-one valued assignments from D to R. Thus $I_P = A_P \cap I_=$ and $Z_P = A_P \cap Z_=$. $\sqcap$

1.2 Sorting Networks

The familiar algorithms for sorting an assignment f make use of two primitive operations: comparison and interchange. For any pair of elements x and y in D, a comparison of x and y determines whether or not $f(x) \leq f(y)$, and an interchange of x and y transforms f into $f' = \pi^{-1} \cdot f$, where π is the permutation (xy). These two primitive operations are combined in the definition of a comparator which can be viewed as a kind of sorting operation on a two element domain. Specifically, a comparator on x and y performs an interchange of x and y if and only if the assignment f has $f(x) > f(y)$. Notice that a comparator on x and y transforms an assignment f into an assignment f' satisfying $f'(x) \leq f'(y)$, so that f'(x) is the minimum of f(x) and f(y) and f'(y) is the maximum of f(x) and f(y).

<u>Definition 1.2.1</u> If x and y are elements of domain D, the <u>comparator</u> $\langle x,y\rangle$ is that operation which transforms any assignment f into the assignment $f' = \langle x,y\rangle(f)$ given by

$$f' = \begin{cases} f & \text{if } f(x) \leq f(y) \\ \pi^{-1} \cdot f & \text{if } f(x) > f(y) \end{cases}$$

where π is the permutation (xy). $\square$

A comparator $\langle x,y \rangle$ may be represented schematically by an arrow from x to y, indicating that the larger of f(x) and f(y) will be assigned to y and the smaller of f(x) and f(y) will be assigned to x. A composition of comparators can be drawn as a network, with locations in D represented by lines drawn from left to right and comparators represented by arrows connecting the lines vertically. For example, if D is the domain $\{a,b,c,d\}$, then the composition of comparators

$$\langle a,b \rangle \cdot \langle d,c \rangle \cdot \langle a,d \rangle \cdot \langle b,c \rangle \cdot \langle b,d \rangle$$

can be represented by any of the networks depicted in Figure 1.2.1

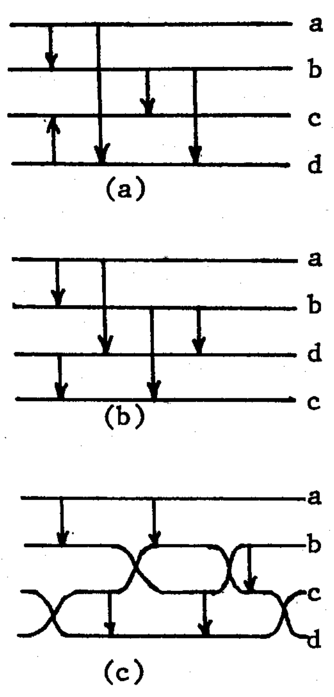


Figure 1.2.1 Examples of comparator networks.

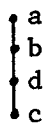
In Figure 1.2.1, the horizontal lines (often called "wires" by analogy with electrical networks) are labeled with the elements of D that they represent. For any labeled network, there is exactly one composition of comparators that the network represents, and for any composition of comparators, there is exactly one labeled network (as long as rearranged versions of the same network do not count as distinct.)

Definition 1.2.2 A comparator network is a composition of comparators. If f is an assignment and C is a comparator network, C(f) will denote the assignment that f is transformed into by the composition of comparators C. If A is a set of assignments, the image of A under a comparator network C is written C(A) and defined to be the set

$$\{C(f) \mid f \in A\} \qquad \square$$

Although a comparator network has been defined abstractly as a composition of functions called comparators, it is possible to implement a comparator as a finite state machine and thus to implement a comparator network as a network of finite state machines. This implementation is discussed more fully in Chapter 2.

Definition 1.2.3 If $C(A_{=}) = A_P$ for some partial order P on D, then C is said to sort with respect to P. If T is a total order on D such that $C(A_{=}) = A_T$, C is called a sorting network (with respect to T). If C is a sorting network with respect to T such that xTy for every comparator <x,y> in the network C, then C is called a standard form sorting network. $\square$

For example, it can be shown that the comparator network described in Figure 1.2.1 is a sorting network; in fact, it is a standard form sorting network with respect to 

.

It is conventional to draw comparator networks with no wire crossings and with the wire labeled x above the wire labeled y iff xTy. If this is done for a standard form sorting network, all of the arrows will point down, so that the arrowheads are redundant and may be omitted. Notice that Figure 1.2.1(b) observes these conventions.

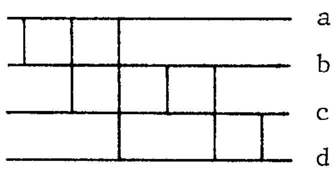
The next theorem is due to [Knuth].

<u>Theorem 1.2.4</u>. For any network C that sorts with respect to a total order T, there exists a standard form network C' that sorts with respect to T and contains the same number of comparators as C. □

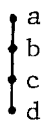
When there is an implicit total order T on the domain D, it will be useful to extend the notion of standard form sorting networks and the conventions for drawing them to comparator networks in general by omitting the arrowheads when xTy for every comparator <x,y> in the network.

In what follows, the usual "higher is greater" convention for ordering diagrams of partial orders is inverted to conform to the conventions for drawing standard form networks. That is, xPy iff x is connected to y by a path going <u>downwards</u> in the ordering diagram for the partial order P. This causes the top-to-bottom arrangement of domain elements to be the same for either the ordering diagram of a total order T or the wires of a standard form network that sorts with respect to T.

Example 1.2.5 The standard form network



is a sorting network with respect to the total order



□

1.3 The Analysis Problem for Sorting Networks

It is often quite difficult to decide whether a comparator network sorts. For example, it is difficult to verify by inspection that the standard form network of Figure 1.3.1 is in fact a sorting network.

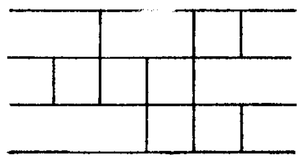


Figure 1.3.1 Sorting network.

This difficulty is reflected in the fact that S(n), the minimum number of comparators in any sorting network on an n-element domain, is unknown for $n > 8$; asymptotically, S(n) is known to be at worst $\sigma(n\log^2 n)$, but might be as small as $\sigma(n\log n)$ [Knuth]. Another manifestation of the same problem is that it is very difficult to design networks which are "good" in the sense that they contain as few comparators as possible; the only systematic design techniques known are based on recursive merges, and give rise to $n\log^2 n$ growth rates in the number of comparators required [VanVoorhis].

There are at least two possible avenues to a better understanding of sorting networks. First, it would be useful to have better criteria for determining whether a comparator network is or is not a sorting network. In particular, a criterion which would lead to an improved upper or lower bound on the number of comparators in a sorting network would of course be very desirable. Second, better techniques for analyzing comparator networks are needed, since a more complete understanding of the capabilities and limitations of comparator networks in general would lead to a better picture of what is going on within sorting networks.

This thesis explores these two avenues. Chapter 2 discusses two criteria, one of them new, for deciding whether a comparator network is a sorting network. Chapter 3 develops two related ways of characterizing the "state of the sort" in terms of sets of assignments that can appear as comparator network outputs. Chapter 4 explores the notion of sorting with respect to a partial order. Finally, Chapter 5 contains a discussion of the results of the previous chapters and recommendations for further research in the area.

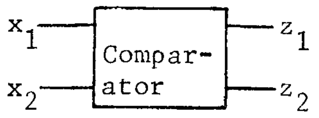
CHAPTER 2

CRITERIA FOR DECIDING WHETHER A NETWORK SORTS

2.1 Comparator Networks as Finite State Machines

This section discusses an implementation of the comparator operation as a finite state machine. In this section, the term "comparator" will refer to the finite state machine implementation rather than to the operation itself, and the term "comparator network" will refer to the implementation of a comparator network as a network of finite state machines. The term "assignment" will refer to an assignment with wires as the domain and sequences of binary digits as the range; each binary sequence implements the binary representation of an integer with the understanding that the most significant bit is first in the sequence.

The state table for a comparator is shown in Figure 2.1.1. The starting state is E, and as long as the upper and lower input symbols x_1 and x_2 agree, the machine remains in state E. As soon as x_1 differs from x_2, the machine enters state L or state G depending on which input digit is 0 and which is 1. State L is entered if x_1 is 0, indicating



x_1	0	0	1	1
x_2	0	1	0	1
E	E, $\begin{matrix}0\\0\end{matrix}$	L, $\begin{matrix}0\\1\end{matrix}$	G, $\begin{matrix}0\\1\end{matrix}$	E, $\begin{matrix}1\\1\end{matrix}$
L	L, $\begin{matrix}0\\0\end{matrix}$	L, $\begin{matrix}0\\1\end{matrix}$	L, $\begin{matrix}1\\0\end{matrix}$	L, $\begin{matrix}1\\1\end{matrix}$
G	G, $\begin{matrix}0\\0\end{matrix}$	G, $\begin{matrix}1\\0\end{matrix}$	G, $\begin{matrix}0\\1\end{matrix}$	G, $\begin{matrix}1\\1\end{matrix}$

Figure 2.1.1. Comparator state table.

that the integer represented on the upper input wire is less than the integer represented on the lower. In state L, $z_1 = x_1$ and $z_2 = x_2$, thus leaving the assignment unchanged. State G is entered if x_2 is 0, indicating that the integer represented on the lower input wire is the smaller of the two; in state G, $z_1 = x_2$ and $z_2 = x_1$ so that the outputs are the transposition of the inputs. If a new assignment is to be input to the comparator, the comparator must be reset to state E.

Since a comparator network is a loop free interconnection of finite state machines, it is itself a finite state machine. A network containing k comparators has 3^k states, some of which may be equivalent or unreachable from the starting state. It will be shown that a comparator network is a sorting network only if it contains a certain number of reachable states.

Definition 2.1.1. Let C be a finite state machine, let f be any assignment, and suppose f takes the machine C from its starting state to state s. Then $C_f(g)$ will denote the output that results when the assignment g is applied to C, starting in state s. $\square$

The notation of Definition 2.1.1 is consistent with that of Definition 1.2.2 under the convention that C(f) is just another way of writing $C_\lambda(f)$, where λ denotes the input sequence of zero length. (Since λ leaves the machine C in its starting state, $C_\lambda(f)$ is the output that results when f is applied to C in its starting state.)

Definition 2.1.2 For any finite state machine C, let $\equiv_c$ denote the binary relation on input sequences defined by

$$f_1 \equiv_c f_2 \Leftrightarrow (\forall g)\ (C_{f_1}(g) = C_{f_2}(g))$$

The relation $\equiv_c$ is an equivalence relation called the Nerode equivalence relation of C. It is a well-known result in automata theory that the blocks of the partition induced by $\equiv_c$ on the set of all input sequences are in one to one correspondence with the reachable states of any reduced machine equivalent to C.

The following theorem is a variation of Bouricius's theorem [Knuth].

Theorem 2.1.3 Let f_1 and f_2 be assignments on domain D and let C be any comparator network on D with $C(f_1) = f_1'$ and $C(f_2) = f_2'$. If

$$(\forall x, y \in D)(f_1(x) \le f_1(y) \Rightarrow f_2(x) \le f_2(y))$$

then

$$(\forall x, y \in D)(f_1'(x) \le f_1'(y) \Rightarrow f_2'(x) \le f_2'(y))$$

That is, if f_2 is an order-homomorph of f_1 then $C(f_2)$ is an order-homomorph of $C(f_1)$.

Proof: The proof is by induction on the number of comparators in C.

Basis: If C has no comparators, $f_1 = f_1'$ and $f_2 = f_2'$ so the statement of the theorem is obviously true.

Induction step: Let the statement of the theorem be true for all networks containing n or fewer comparators, and let C' be a network of n+1 comparators. C' can be viewed as the composition of a network C of n comparators and a single comparator $\langle a,b\rangle$. Let f_1 and f_2 be any two assignments on D with $C(f_1) = f_1'$ and $C(f_2) = f_2'$, and suppose that

$$(\forall x, y \in D)(f_1(x) \le f_1(y) \Rightarrow f_2(x) \le f_2(y)).$$

Since C has n comparators, the induction hypothesis guarantees that

$$(\forall x, y \in D)(f_1'(x) \le f_1'(y) \Rightarrow f_2'(x) \le f_2'(y)).$$

Letting $\langle a,b\rangle(f_1') = f_1''$ and $\langle a,b\rangle(f_2') = f_2''$, it is clear that $f_1' = f_1''$ and $f_2' = f_2''$ as long as $f_1'(a) \le f_1'(b)$. If $f_1'(a) > f_1'(b)$, then $f_1'(b) \le f_1'(a)$ and $f_2'(b) \le f_2'(a)$, so that $f_1'' = \pi^{-1} \cdot f_1'$ and $f_2'' = \pi^{-1} \cdot f_2'$ where π is the permutation (ab). In any event,

$$(\forall x, y \in D)(f_1''(x) \le f_1''(y) \Rightarrow f_2''(x) \le f_2''(y)). \qquad \square$$

Corollary 2.1.4 If f_1 and f_2 are order-isomorphic assignments on D, i.e. if

$$(\forall x, y \in D)(f_1(x) \le f_1(y) \Leftrightarrow f_2(x) \le f_2(y))$$

then for any comparator network C on D, f_1 and f_2 leave every comparator of C in the same state.

Proof: For any comparator $<a,b>$ in C, apply the theorem to the portion of the network to the left of $<a,b>$. The assignments f_1' and f_2' that appear at the input of $<a,b>$ when C is applied to f_1 and f_2, respectively, satisfy

$$f_1'(a) \le f_1'(b) \Leftrightarrow f_2'(a) \le f_2'(b)$$

It is easy to verify that

$$\begin{aligned} f_1'(a) < f_1'(b) &\Leftrightarrow f_2'(a) < f_2'(b) \\ f_1'(a) = f_1'(b) &\Leftrightarrow f_2'(a) = f_2'(b) \\ f_1'(a) > f_1'(b) &\Leftrightarrow f_2'(a) > f_2'(b) \end{aligned}$$

so that f_1' and f_2' leave $<a,b>$ in the same state. □

Definition 2.1.5 Let $\equiv$ denote the binary relation on $A_=$ defined by

$$f_1 \equiv f_2 \Leftrightarrow (\forall x, y \in D)(f_1(x) \le f_1(y) \Leftrightarrow f_2(x) \le f_2(y))$$

That is, $f_1 \equiv f_2$ iff f_1 and f_2 are order isomorphic. □

The binary relation $\equiv$ is an equivalence relation on $A_=$ whose equivalence classes (order isomorphism classes) form a partition of $A_=$. It will be useful to represent each block of the partition by a particular element of the block.

Definition 2.1.6 For every assignment f on the domain D the assignment $\hat{f}$ is defined as follows:

$$\hat{f}(x) = |\{f(z) | f(z) < f(x)\}|$$

That is, $\hat{f}(x)$ is the number of values f(z) in the range of f that are less than f(x). □

Example 2.1.7 If f is the assignment on $D = \{a,b,c,d\}$ defined by $f(a) = 3$, $f(b) = 4$, and $f(c) = f(d) = 1$, then $\hat{f}$ is the assignment defined by $\hat{f}(a) = |\{1\}| = 1$, $\hat{f}(b) = |\{1,3\}| = 2$, and $\hat{f}(c) = \hat{f}(d) = |\{\}| = 0$. ◻

Theorem 2.1.8 For any assignment f in $A_{=}$, $\hat{f} \equiv f$.

Proof: Suppose $f(x) \le f(y)$. Then

$$\{f(z) | f(z) < f(x)\} \subseteq \{f(z) | f(z) < f(y)\}$$

so $\hat{f}(x) \le \hat{f}(y)$. Now suppose $f(x) \le f(y)$ is false, so that $f(x) > f(y)$; then

$$\{f(z) | f(z) < f(y)\} \subset \{f(z) | f(z) < f(x)\}$$

and $\hat{f}(x) > \hat{f}(y)$, so $\hat{f}(x) \le \hat{f}(y)$ is false. ◻

Theorem 2.1.9 For any two assignments f_1 and f_2 in $A_{=}$, $f_1 \equiv f_2 \Rightarrow \hat{f}_1 = \hat{f}_2$.

Proof: Let $f_1 \equiv f_2$. Then for all x, $f_1(z) < f_1(x)$ iff $f_2(z) < f_2(x)$ and $f_1(z) = f_1(z')$ iff $f_2(z) = f_2(z')$, from which it follows that $|\{f_1(z) | f_1(z) < f_1(x)\}| = |\{f_2(z) | f_2(z) < f_2(x)\}|$ and $\hat{f}_1 = \hat{f}_2$. ◻

Theorems 2.1.8 and 2.1.9 imply that every block of the partition of $A_{=}$ induced by $\equiv$ contains exactly one assignment of the form $\hat{f}$. It will now be argued that if C is a sorting network, $\equiv_c$ is the same equivalence relation as $\equiv$.

Theorem 2.1.10 If C is a sorting network on an n-element domain D, then for all assignments f_1 and f_2 in $A_=$, $f_1 \equiv_c f_2 \Leftrightarrow f_1 \equiv f_2$.

Proof: In view of Corollary 2.1.4, $f_1 \equiv f_2 \Rightarrow f_1 \equiv_c f_2$. If $f_1 \equiv_c f_2$, then since $f_1 \equiv \hat{f}_1$ and $f_2 \equiv \hat{f}_2$, $f_1 \equiv_c \hat{f}_1$ and $f_2 \equiv_c \hat{f}_2$, i.e. $\hat{f}_1 \equiv_c \hat{f}_2$. Since $\hat{f}_1 = \hat{f}_2$ implies $f_1 \equiv f_2$, it remains to show that $\hat{f}_1 \equiv_c \hat{f}_2$ implies $\hat{f}_1 = \hat{f}_2$. It will be convenient to let the domain D be the set $\{0,1,2,\ldots n-1\}$.

Let $\hat{f}_1 \neq \hat{f}_2$, so that $\hat{f}_1(i) \neq \hat{f}_2(i)$ for some i. The set

$$\{\min(\hat{f}_1(i),\ \hat{f}_2(i)) \mid \hat{f}_1(i) \neq \hat{f}_2(i)\}$$

is nonempty; let j be such that $\min(\hat{f}_1(j), \hat{f}_2(j))$ is the least element of this set and $f_1(j) \neq f_2(j)$; without loss of generality, let $\hat{f}_1(j) < \hat{f}_2(j)$. Let k denote the number of locations i in D such that $\hat{f}_1(i) < \hat{f}_1(j)$, and define the function g as follows:

$$g(i) \begin{cases} 0 & \text{if } i = j \\ i+1 & \text{otherwise} \end{cases}$$

Now consider $C_{\hat{f}_1}(g) = h_1$ and $C_{\hat{f}_2}(g) = h_2$. It will be argued that $h_1(k) = 0$ and $h_2(k) \neq 0$. First, since there are k locations i with $\hat{f}_1(i)$ less than $\hat{f}_1(j)$, $C(\hat{f}_1) = \hat{f}_1'$ is an assignment with $\hat{f}_1'(k) = \hat{f}_1(j)$ because C is a sorting network. Evicently, $h_1(k)$ is the smallest of the values g(i) such that $\hat{f}_1(i) = \hat{f}_1(j)$; since g(j) is the smallest possible such value, $h_1(k) = g(j) = 0$.

Now consider $h_2(k)$. $h_2(k) = 0$ only if there are exactly k locations i with $\hat{f}_2(i) < \hat{f}_2(j)$. But since $\hat{f}_1(i) = \hat{f}_2(i)$ for every i such that $\hat{f}_1(i) < \hat{f}_1(j)$ by definition of j, there are exactly k locations i such that $\hat{f}_2(i) < \hat{f}_1(j)$. This implies that for no location i is it the case that $\hat{f}_1(j) \leq \hat{f}_2(i) < \hat{f}_2(j)$. In particular, there can be no location i with $\hat{f}_2(i) = \hat{f}_2(j)-1$. Since $\hat{f}_2(j)$ is the number of elements in the set $\{f_2(z) | f_2(z) < f_2(j)\}$, this is a contradiction. Therefore $h_2(k) \neq 0$ and $h_1 \neq h_2$, implying $\hat{f}_1 \equiv_c \hat{f}_2$ is false. □

Example 2.1.11 This example illustrates the proof of Theorem 2.1.10. The assignments $\hat{f}_1$, $\hat{f}_2$, $C(\hat{f}_1) = \hat{f}_1'$, $C(\hat{f}_2) = \hat{f}_2'$, g, h_1, and h_2 are given in the table below.

i	$\hat{f}_1(i)$	$\hat{f}_2(i)$	g(i)	$\hat{f}_1'(i)$	$h_1(i)$	$\hat{f}_2'(i)$	$h_2(i)$
0	2	1	1	0	2	0	2
1	0	0	2	0	5	0	5
2	1	1	3	0	8	0	8
3	3	2	4	1	0	1	1
4	0	0	5	1	3	1	3
5	1	3	0	2	1	2	4
6	2	2	7	2	7	2	7
7	0	0	8	3	4	3	0

First, notice that the smallest element of $\{\min(\hat{f}_1(i), \hat{f}_2(i)) | \hat{f}_1(i) \neq \hat{f}_2(i)\}$ is 1; since $\hat{f}_1(5) = 1$ and $\hat{f}_2(5) = 3$, set $j = 5$. (The function g given in the table was chosen for this location j, although $j = 0$ would have worked equally well.) Since $\hat{f}_1(i) < \hat{f}_1(5)$ for $i = 1,4$, and 7, set $k = 3$. Notice that $h_1(3) = 0$ because $g\ (5) = 0$, but because $\hat{f}_2(i) < \hat{f}_2(5)$ for 7 locations i and 7 is greater than 3, $h_2(3) \neq 0$; in fact, $h_2(7) = 0$. ⊓

Corollary 2.1.12 No two reachable states in a sorting network C are equivalent.

Proof: Suppose f_1 and f_2 leave C in equivalent states, so that $f_1 \equiv_c f_2$; by Theorem 2.1.10, $f_1 \equiv f_2$; finally by Corollary 2.1.4, f_1 and f_2 leave C in the same state. ⊓

Corollary 2.1.13 A sorting network C on an n-element domain has

$$\sigma(n) = \sum_{k=0}^{n} k!\ \left\{ {n \atop k} \right\}$$

reachable states, where $\left\{ {n \atop k} \right\}$ denotes a Stirling number of the second kind.

Proof: It suffices to count the number of distinct functions of the form $\hat{f}$ for a domain of n elements. In fact, the number of functions $\hat{f}$ is equal to the number of totally ordered partitions on an n element domain D. To see this, let Σ be such a partition, totally ordered by T, and for any element $x \in D$ let $\hat{f}(x)$ be the number of blocks of Σ that are T-less than the block containing x. It is not hard to verify that there is precisely one ordered partition corresponding to a given $\hat{f}$ and vice versa. Since $\left\{ {n \atop k} \right\}$ is the number of unordered partitions of an n element set containing k blocks, and since there are k! ways of ordering every such partition, $\sigma(n)$ is the number of totally ordered partitions on an n element domain and hence the number of functions of the form $\hat{f}$. □

It can also be shown that any comparator network that has $\sigma(n)$ reachable states has the property that the reachable submachine of the network is isomorphic to the reachable submachine of a sorting network with respect to state transitions.

2.2 The Zero-one Principle

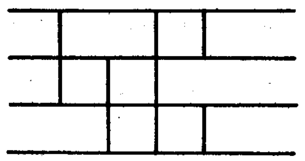
Where as it might be thought that some comparator network could successfully sort all zero-one valued assignments and yet fail to sort a more "complex" assignment, this cannot in fact occur; Theorem 2.2.1 shows that a test to see if all zero-one valued assignments are sorted by the network is conclusive. A proof may be found in [Knuth].

Theorem 2.2.1 (zero-one principle) A comparator network sorts every assignment in $A_=$ if it sorts every assignment in $Z_=$. □

The zero one principle is very useful for deciding whether or not a comparator network sorts.

It is the purpose of this section to explore the question "for which subsets of $Z_=$ do there exist comparator networks that sort precisely the assignments in those subsets". For example, it is possible for a comparator network to sort every zero-one valued assignment but one? the answer is yes. To prove this, a lemma will be needed. The proof is straightforward.

<u>Lemma 2.2.2</u> The network N depicted below sorts every element of $Z_=$



except the assignment $\begin{matrix}0\\1\\0\\1\end{matrix}$, and $N\begin{pmatrix}0\\1\\0\\1\end{pmatrix} = \begin{matrix}0\\1\\0\\1\end{matrix}$. □

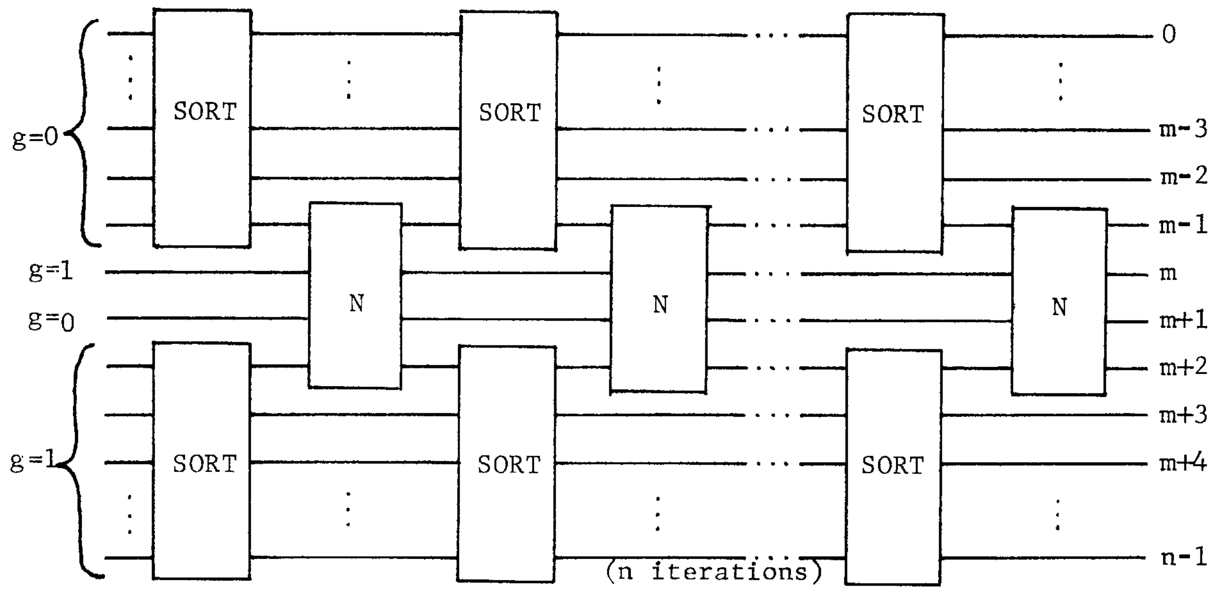
Now for the theorem itself.

<u>Theorem 2.2.3</u> If g is any assignment in $Z_=$ except the constant zero or constant one functions, there exists a comparator network which sorts every element of $Z_=$ except g with respect to some total order T.

<u>Proof</u>: Let the domain D be the set {0,1,2,...n-1}, and let g be an assignment on D of the form

$$\begin{array}{r l} \begin{array}{c}m\\ \text{locations}\\ (0 \text{ to } m-1)\end{array} & \left\{\begin{array}{c}0\\ \vdots\\ 0\\ 0\\ 0\end{array}\right. \\ & \;\;1\\ & \;\;0 \\ \begin{array}{c}n-m-2\\ \text{locations}\\ (m+2 \text{ to } n-1)\end{array} & \left\{\begin{array}{c}1\\ 1\\ 1\\ \vdots \\ 1\end{array}\right.\end{array}$$

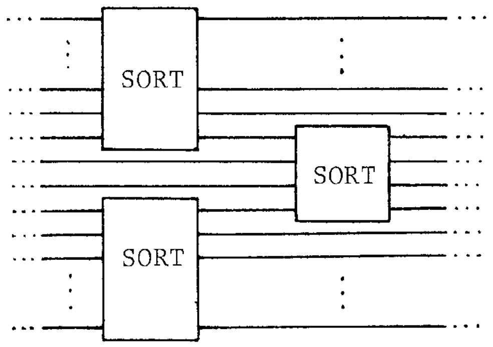
where $0 \le m \le n-2$. Now let g be applied to the standard form network C shown below. The boxes marked "SORT" are sorting networks, and the boxes marked "N" are networks of the kind shown in Lemma 2.2.2.



It is easy to verify that C(g) = g is not sorted with respect to $\le$. Now let $h \neq g$ be any other assignment in $Z_{=}$. If h is applied to C, then at least one of the following four things must be true:

1. There is at least one 1 on input wires 0 through m-1;
2. There is a 0 on input wire m;
3. There is a 1 on input wire m+1;
4. There is at least one 0 on input wires m+2 through n-1.

In any event, the leftmost pair of sorting networks in C transform h into an assignment such that the leftmost N network in C does not receive the pattern $\begin{matrix}0\\1\\0\\1\end{matrix}$ at its inputs. It follows from Lemma 2.2.2 that whatever this assignment may be, it is transformed by the leftmost N network into an assignment which is sorted on the wires m-1, m, m+1, and m+2; in particular, it cannot be the case that wires m and m+1 are assigned 1 and 0, respectively, at the output of the leftmost N network. This means chat $\begin{matrix}0\\1\\0\\1\end{matrix}$ cannot appear at the inputs of the next N network in C, or, inductively, at the input of <u>any</u> subsequent N network in C. In fact, every N network in C could be replaced by a four input standard form sorting network with no effect on the ultimate network output C(h). After this has been done, it is not hard to see that C(h) is sorted with respect to ≤; just verify that each iteration of the form



decreases the distance between the uppermost one and the lowermost zero by at least one in propagating the assignment, and use the fact that there are n such iterations in C. For the degenerate cases $m = 0$ and $m = n-2$, it will be necessary to remove the top (respectively bottom) wire from every N network together with all comparators touching that wire.

Now let g' be an arbitrary nonconstant assignment in $Z_=$, and let π be a permutation on D such that $g = \pi \cdot g'$ is an assignment of the form

$$\begin{array}{r l} \begin{array}{c} m \\ \text{locations} \end{array} & \left\{\begin{array}{c}0\\ \vdots\\ 0\\ 0\\ 0\end{array}\right. \\ & \;1 \\ & \;0 \\ \begin{array}{c} n-m-2 \\ \text{locations} \end{array} & \left\{\begin{array}{c}1\\ 1\\ 1\\ \vdots\\ 1\end{array}\right. \end{array}$$

If C denotes the network that sorts every member of $Z_=$ except g, let C' denote the network obtained by replacing every comparator $<x,y>$ in C by the comparator $<\pi(x), \pi(y)>$. Since $g(x) \le g(y)$ iff $g'(\pi(x)) \le g'(\pi(y))$, C' sorts every assignment in $Z_=$ with respect to the total order T that satisfies

$$(\forall x, y \in D) \; (\pi(x) T \pi(y) \Leftrightarrow x \le y)$$

i.e. the total order defined by

$$(\forall u, v \in D)(u T v \Leftrightarrow \pi^{-1}(u) \le \pi^{-1}(v))$$

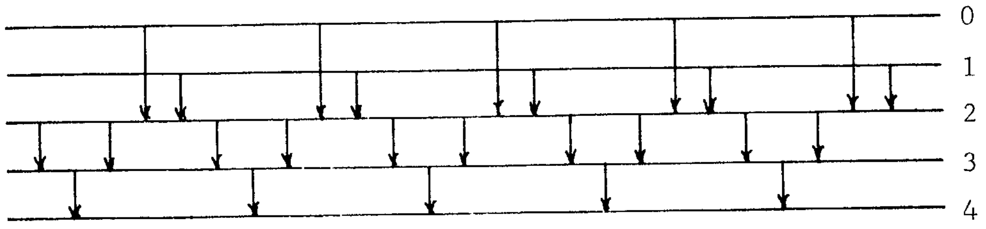
but fails to sort the assignment g'. $\square$

An example will be useful to clarify the construction of Theorem 2.2.3.

Example 2.2.4. It is desired to construct a network which sorts every assignment but the assignment g' on D = {0,1,2,3,4} defined by

$$g'(x) = \begin{cases} 0 & \text{if } x = 3 \\ 1 & \text{otherwise} \end{cases}$$

Now the network C shown below

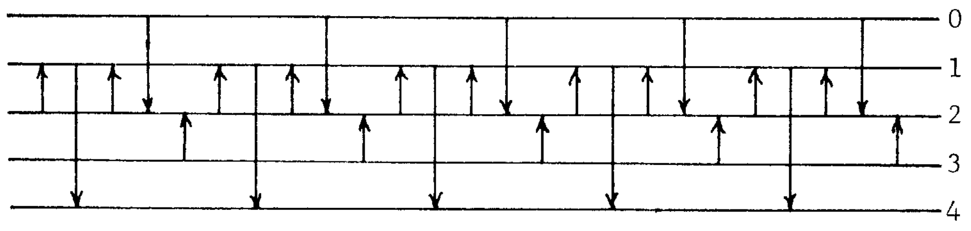


sorts every assignment but g, where g is defined by

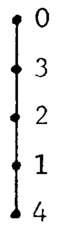
$$g(x) = \begin{cases} 0 & \text{if } x = 1 \\ 1 & \text{otherwise} \end{cases}$$

(Notice that since there is only one wire for which g(x) = 0, the top wire of the network N has been removed together with all comparators touching that wire.)

Since π = (13) is a permutation such that $g = \pi \cdot g'$, the network C' shown below



sorts every element of $Z_{=}$ except g' with respect to

T = 

<u>Corollary 2.2.5</u> In verifying that an n-input comparator network sorts by means of the zero-one principle, it is necessary to try 2^n-2 inputs, namely all the nonconstant zero-one valued assignments. $\sqcap$

Theorem 2.2.3 could be modifed to deal with standard form networks, which of course must sort at least those zero-one valued assignments that are already sorted.

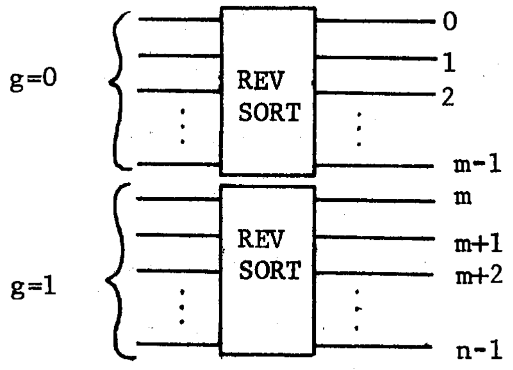
Among other things, Theorem 2.2.3 indicates that all nontrivial statements of the form "any comparator network that sorts every element of this set of zero-one valued assignments also sorts that zero-one valued assignment" are false. The next theorem, analogously, denies the possibility of similar statements about assignments <u>not</u> sorted.

<u>Theorem 2.2.6</u> For any zero-one valued assignment g, there exists a total order T and a comparator network C which sorts only that assignment (and the two constant assignments) with respect to T.

<u>Proof</u>: Let g be an assignment on D = {0,1,2,...n-1} of the form

$$\begin{array}{rl} \begin{array}{c}\text{m}\\ \text{locations}\end{array} & \left\{\begin{array}{l}0\\0\\0\\\vdots\\0\end{array}\right. \\ \begin{array}{c}\text{n-m}\\ \text{locations}\end{array} & \left\{\begin{array}{l}1\\1\\1\\\vdots\\1\end{array}\right. \end{array}$$

where $0 \le m \le n$. Now let g be applied to the network C shown below. The boxes marked "REV SORT" are sorting networks which sort in reverse, i.e. with respect to the total order $\ge$.



Clearly C(g) = g is sorted with respect to $\le$, as are the constant zero and constant one assignments, but no other assignment in $Z_=$ is sorted with respect to $\le$. Now let g' be any assignment in $Z_=$, and let π be a permutation on D such that $g = \pi \cdot g'$ is an assignment of the form

$$\begin{array}{r l}\left.\begin{array}{r}\text{m}\\ \text{locations}\end{array}\right\{ & \begin{array}{c}0\\0\\0\\ \vdots\\0\end{array}\\ \left.\begin{array}{r}\text{n-m}\\ \text{locations}\end{array}\right\{ & \begin{array}{c}1\\1\\1\\ \vdots\\1\end{array}\end{array}$$

A transformation of the comparators in C analogous to that performed in Theorem 2.2.3 establishes the theorem. □

There are certainly some sets of zero-one valued assignments for which there can be no comparator network sorting exactly the members of that set.

<u>Example 2.2.5</u> No comparator network containing one or more comparators can sort each of the assignments.

$$\begin{array}{ccccc} 0 & 0 & 0 & & 1\\ \vdots & \vdots & \vdots & & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ 0 & 0 & 1 & & 1\\ 0, & 1, & 1 & & 1 \end{array}$$

without sorting some other assignment.

Proof: The leftmost comparator in the network will receive a zero at one input and a one at the other for some assignment in the list. The assignment which results in the opposite arrangement of the zero and the one at the leftmost comparator input must also be sorted, but it is not in the list because interchanging a zero and a one in any assignment in the list will result in an assignment not in the list. $\square$

CHAPTER 3

COMPARATOR NETWORK OUTPUT BEHAVIOR

3.1 Output Characterizations

The set of assignments that can appear at the output of a comparator network is sufficient to determine which networks can be concatenated to the network to make the combination a sorting network. Thus it is often useful to know the assignments that can appear at the output of a given comparator network. To the extent that sets of output assignments can be represented succinctly and manipulated easily, the problems of designing and analyzing sorting networks will become easier. In this chapter, two different kinds of assignments will be considered: injective assignments and zero-one valued assignments. These two kinds of assignments are related by the notion of threshold.

Definition 3.1.1 If f is any assignment from D to R then $\theta(f)$, the set of thresholds of f, is defined by

$$\theta(f) = \{g:D \to \{0,1\} \mid (\exists r \in R)(\forall x \in D)$$
$$(g(x) = 1 \text{ iff } r \le f(x)\}$$

If A is any set of assignments from D to R, then

$$\theta(A) = \bigcup_{f \in A} \theta(f)$$

is the *set of thresholds of A*. $\square$

Example 3.1.2 The set of thresholds of the assignment $f: \{a,b,c,d\} \to \{0,1,2,3\}$ defined by $f(a) = 0$, $f(b) = f(c) = 1$, $f(d) = 2$ is the set $\{g_0, g_1, g_2, g_3\}$, where the functions g_i are given in the table below.

x	f(x)	$g_0(x)$	$g_1(x)$	$g_2(x)$	$g_3(x)$
a	0	1	0	0	0
b	1	1	1	0	0
c	1	1	1	0	0
d	2	1	1	1	0

$\square$

Notice that an assignment f is sorted iff all of its thresholds are sorted.

The following theorem is a close relative of the zero-one principle.

Theorem 3.1.3 For any network C and any assignment f, $C(\theta(f)) = \theta(C(f))$.

Proof: Let $g' \in C(\theta(f))$, so that $g' = C(g)$ for some $g \in \theta(f)$. Since $f(x) \le f(y) \Rightarrow g(x) \le g(y)$ for all x and y, Theorem 2.1.3 guarantees that $f'(x) \le f'(y) \Rightarrow g'(x) \le g'(y)$. If $g'(x) = 0$ for all x, certainly $g' \in \theta(f')$; otherwise, set

$$r' = \min_{x \in D} \{f'(x) | g'(x) = 1\}$$

and let z be any element of D for which $f'(z) = r'$, hence for which $g'(z) = 1$. Now for all x, if $r' \le f'(x)$, i.e. if $f'(z) \le f'(x)$, then $1 = g'(z) \le g'(x)$ and $g'(x) = 1$. Conversely, if $g'(x) = 1$, then $r' \le f'(x)$ by definition of r'. It follows that $g' \in \theta(f')$, and so $C(\theta(f)) \subseteq \theta(C(f))$. Now, every element g in $\theta(f)$ is characterized uniquely by the number of elements in D mapped to zero by g. Since C merely permutes the domain of each $g \in \theta(f)$, $|C(\theta(f))| = |\theta(f)|$; since C applied to f just permutes the domain of f, $|\theta(C(f))| =$

$|\theta(f)|$. Hence $C(\theta(f)) = \theta(C(f))$. □

The following corollary illustrates the use of Theorem 3.1.3.

Corollary 3.1.4 For any two networks C and C', the following statements are equivalent:

1. C' sorts every assignment in $C(Z_=)$, the set of zero-one valued assignments that can appear at the output of C.

2. C concatenated with C' is a sorting network.

3. C' sorts every assignment in $C(I_=)$, the set of injective assignments that can appear at the output of C.

Proof: 1 implies 2 because of the zero-one principle, and 2 implies 3 by virtue of the definition of a sorting network. It remains to show that 3 implies 1. Let g' be any zero-one valued assignment that can appear at the output of C. Then $g' = C(g)$ for some g. Now, $g \in \theta(f)$ for some injective assignment f; letting $C(f) = f'$, $g' \in \theta(f')$ by Theorem 3.1.3. Since C' sorts every injective assignment in $C(I_=)$, $C'(f')$ is sorted and so is every element of $\theta(C'(f'))$. Since $\theta(C'(f')) = C'(\theta(f'))$, every element of $\theta(f')$ (including g') is sorted by C'. □

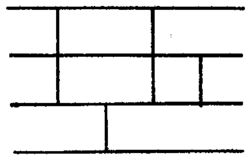
Corollary 3.1.4 guarantees that either the zero-one valued output assignments $C(Z_=)$ or the injective output assignments $C(I_=)$ adequately characterize a network C for purposes of designing a network C' so that C concatenated with C' is a sorting network. On the other hand, if the network C' is given and it is desired to design a network C so that C concatenated with C' is a sorting network, then the network C' is characterized for this purpose by those assignments that it sorts. In the light of Theorem 3.1.3, it is not surprising that the zero-one valued assignments sorted by C' characterize the injective assignments sorted by C'.

Theorem 3.1.5 If I is the set of injective assignments sorted by the network C' then for every injective f, $\theta(f) \subseteq \theta(I) \Rightarrow f \in I$.

Proof: Let $\theta(f) \subseteq \theta(I)$ with $f \notin I$. Then $f' = C'(f)$ is not sorted, so there is a zero-one valued assignment $g' \in \theta(f')$ which is not sorted. Since $g' \in \theta(C'(f))$, $g' \in C'(\theta(f))$ and so $g' = C'(g)$ for some $g \in \theta(f)$. Since $\theta(f) \subseteq \theta(I)$, there exists an injective h in I with $g \in \theta(h)$. Since $h \in I$, $C'(h)$ is sorted and therefore every element of $\theta(C'(h))$ is sorted. This implies $g' \in C'(\theta(h))$ is sorted, a contradiction. □

The example below shows that C' may in fact have to sort more injective assignments than those in $C(I_{=})$ if C concatenated with C' is to be a sorting network.

Example 3.1.6 The network



transforms the set $I_=$ of injective assignments with range $\{1,2,3,4\}$ into the set

$$C(I_=) = \left\{ \begin{matrix} 1 & 2 & 1 \\ 2 & 1 & 2 \\ 3 & 3 & 4 \\ 4, & 4, & 3 \end{matrix} \right\}$$

and transforms $Z_=$ into the set

$$C(Z_=) = \left\{ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0, & 1, & 0, & 1, & 1, & 1, & 1 \end{matrix} \right\}$$

Since $\theta \begin{pmatrix} 2 \\ 1 \\ 4 \\ 3 \end{pmatrix} \subseteq C(Z_=)$ and since any comparator network C' that sorts every element of $C(I_=)$ must also sort every element of $C(Z_=)$, such a network C' sorts $\begin{matrix} 2 \\ 1 \\ 4 \\ 3 \end{matrix}$. $\square$

This example suggests that zero-one valued assignments may be more useful than injective assignments for output representation. This hypothesis will be reinforced by the results of section 3.4; the next two sections will be devoted to a more detailed examination of the properties of these two kinds of output assignments.

3.2 Zero-one valued output assignment

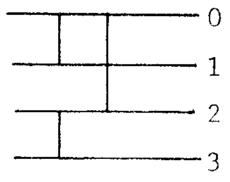
Any set Z of zero-one valued assignments can be described by its characteristic function, which in turn can be described by a Boolean expression. For convenience, let the domain of the assignments be $D = \{0,1,\ldots n-1\}$.

<u>Definition 3.2.1</u> If $Z \subseteq Z_=$ is a set of zero-one valued assignments on domain $D = \{0,1,\ldots n-1\}$, then the <u>characteristic function of Z</u> is the function k from $Z_=$ to $\{0,1\}$ defined by

$$k(g) = \begin{cases} 1 & \text{if } g \in Z \\ 0 & \text{if } g \notin Z \end{cases}$$

$\sqcap$

Since there are 2^n assignments g in $Z_=$, each of which is a function from D to $\{0,1\}$, it is possible to represent a characteristic function k by a Karnaugh map or a Boolean expression in the n variables x_0, x_1, ... x_{n-1}. For example, the network



has the set of output assignments $C(Z_=)$ described by the table below.

x								
0	0	0	0	0	0	0	0	1
1	0	0	1	0	1	1	1	1
2	0	0	0	1	1	0	1	1
3	0	1	0	1	0	1	1	1

The Karnaugh map for the characteristic function of $C(Z_=)$ is

x_0x_1 \ x_2x_3	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	0	0	1	0
10	0	0	0	0

and this function can be described by the expression $x_0'x_1 + x_0'x_2' + x_0'x_3 + x_1x_2x_3$. In this fashion, a Boolean expression can describe any set of zero-one valued assignments, and in particular, the set of zero-one valued outputs $C(\underline{Z})$ of a comparator network C.

A comparator $<i,j>$ concatenated to the output of a network C results in a new network $C' = C\cdot<i,j>$, a new set of zero-one valued output assignments $C'(\underline{Z})$ and hence a new characteristic function described by a new expression. The following definition will be useful in describing this phenomenon.

Definition 3.2.2 If E is a Boolean expression in the variables $x_0, x_1, \ldots x_{n-1}$, then $E/_{(x_i,x_j)}$ is the expression obtained by replacing every occurence of x_i in E by x_j and every occurence of x_j in E by x_i. □

The theorem below describes the effect of a comparator $<i,j>$ on a set Z of zero-one valued assignments. This is done by giving an expression $\hat{E}$ for the characteristic function of $<i,j>$ (Z) in terms of i,j, and any expression E for the characteristic function of Z.

Theorem 3.2.3 If Z is a set of zero-one valued assignments on the domain D = {0,1,...n-1} and E is a Boolean expression for the characteristic function of Z in the variables $x_0, x_1, \ldots, x_{n-1}$, then the comparator <i,j> transforms Z into a set whose characteristic function is described by the expression $\hat{E}$, where

$$\hat{E} = Ex_i' + E/_{(x_i,x_j)}\, x_j$$

Proof: Every assignment $g \in Z$ can be classified as to whether g(i) = 0 or g(i) = 1. Let Z_0 and Z_1 partition Z according to g(i):

$$Z_0 = \{g \in Z | g(i) = 0\}$$
$$Z_1 = \{g \in Z | g(i) = 1\}$$

and let E_0 be the expression Ex_i' and E_1 the expression Ex_i. E_0 is an expression for the characteristic function of Z_0 because $g \in Z_0$ iff $g \in Z$ and g(i) = 0; similarly, E_1 is an expression for the characteristic function of Z_1. Since $Z = Z_0 \cup Z_1$, the expression $E_0 + E_1$ is an expression for the characteristic function of Z; moreover, the effect of the comparator <i,j> on Z can be determined by its effects on Z_0 and Z_1. In particular, the comparator has no effect on Z_0, since g(i) = 0 for every $g \in Z_0$, but the comparator interchanges the values of g(i) and g(j) for every $g \in Z_1$ since g(i) = 1 for these g. It follows that

$$\hat{E} = E_0 + E_1/_{(x_i,x_j)} = Ex_i' + (Ex_i)/_{(x_i,x_j)}$$

or

$$\hat{E} = Ex_i' + E/_{(x_i,x_j)}x_j$$

is an expression for the characteristic function of the set $\langle i,j \rangle$ (Z). □

Since convenient representations for sets of zero-one valued assignments are desirable, let us explore representations for Boolean expressions. The product of sums form for expressions seems to be most useful because it provides a convenient way of representing the kinds of statements used to describe consistency. First, a theorem about the form of any product of sums expression for a set of output assignments.

<u>Theorem 3.2.4</u> If C is any comparator network and E is any product of sums expression for the characteristic function of $C(Z_=)$, then every factor in E must contain at least one complemented variable and at least one uncomplemented variable.

<u>Proof</u>: It is clear that $C(Z_=)$ must contain both the constant zero assignment (the assignment that is zero for every location in D) and the constant one assignment. Now suppose some factor in E contains no complemented variable. Then the constant zero assignment could not appear in $C(Z_=)$, a contradiction. Similarly, every factor in E must contain an uncomplemented variable if the constant one assignment is to appear in $C(Z_=)$. □

<u>Corollary 3.2.5</u> The set of zero-one valued output assignments $C(Z_=)$ for any comparator network C has a characteristic function which can be described by an expression of the form

$$E = \prod_{k=1}^{m} \left[\left(\prod_{i \in A_k} x_i \right)' + \left(\sum_{j \in B_k} x_j \right) \right]$$

Proof: Write the kth factor in a product of sums form for E as $\sum_{i \in A_k} x_i' + \sum_{j \in B_k} x_j$; theorem 3.2.4 guarantees that the sets A_k and B_k are nonempty. The result then follows from DeMorgan's law. $\square$

When the set $C(Z_=)$ of zero-one valued output assignments consists precisely of those assignments consistent with a certain partial order P, then a particularly simple form for E is possible, based on the covering relation for P.

Definition 3.2.6 If P is a partial order on D, then the covering relation for P is the relation $\bar{P}$ on D defined by

$$i\bar{P}j \Leftrightarrow iPj \wedge i \neq j \wedge (\forall k)$$
$$((iPk \wedge kPj) \Rightarrow (k=i \vee k=j))$$ $\square$

The covering relation $\bar{P}$ for a partial order P is irreflexive, antisymmetric, and intransitive, and any irreflexive, antisymmetric, and intransitive relation $\bar{P}$ is the covering relation for some partial order P (namely the reflexive transitive closure of $\bar{P}$).

Theorem 3.2.7 Let C be a comparator network. The characteristic function for $C(Z_=)$ has an expression of the form

$$E_{\bar{P}} = \prod_{(i,j)\in\bar{P}} (x_i' + x_j)$$

where $\bar{P}$ is irreflexive, antisymmetric, and intransitive, if and only if $C(Z_=) = Z_P$ for some partial order P with $\bar{P}$ the covering relation for P.

Proof: It will be shown that for any partial order P, $E_{\bar{P}}$ is an expression for the characteristic function of Z_P; since every partial order has a unique covering relation and every irreflexive, antisymmetric, intransitive relation is the covering relation for some partial order, this will prove the theorem. So let P be a partial order on D, with $\bar{P}$ the covering relation for P, and suppose $g \in Z_P$. Then $iPj \Rightarrow g(i) \le g(j)$, and since $\bar{P} \subseteq P$, $i\bar{P}j \Rightarrow g(i) \le g(j)$. This means that $g(i) = 0$ or $g(j) = 1$ for all ordered pairs $(i,j) \in \bar{P}$, so that the expression $E_{\bar{P}}$ evaluates to 1 (the characteristic function evaluates to 1) for g. Now suppose $E_{\bar{P}}$ evaluates to 1 for some $g \in Z_=$, so that $g(i) = 0$ or $g(j) = 1$ for all $(i,j) \in \bar{P}$. This means that $i\bar{P}j \Rightarrow g(i) \le g(j)$. Now let iPj. Since P is the reflexive transitive closure of $\bar{P}$, either $i=j$, and $g(i) \le g(j)$, or there exists a chain $i = a_0, a_1, a_2, \ldots a_r = j$ in P with $a_0\bar{P}a_1$, $a_1\bar{P}a_2, \ldots a_{r-1}\bar{P}a_r$. In the latter case it follows that $g(a_0) \le g(a_1) \le g(a_2) \le \ldots \le g(a_r)$, so that $g(i) \le g(j)$. Having shown that $iPj \Rightarrow g(i) \le g(j)$, we conclude that $g \in Z_P$. ⊓

Theorem 3.2.8 provides a generalization of Theorem 3.2.7.

Theorem 3.2.8 Let C be a comparator network on D, and let $R \subseteq D \times D$ be a binary relation such that

$$E_R = \prod_{(i,j)\in R} (x_i' + x_j)$$

is an expression for the characteristic function of $C(Z_=)$. Then there exists an irreflexive, antisymmetric, and intransitive relation $\bar{R} \subseteq R$ such that

$$E_{\bar{R}} = \prod_{(i,j)\in \bar{R}} (x_i' + x_j)$$

is also an expression for the characteristic function of $C(Z_=)$.

Proof: It will be shown that R is antisymmetric. Given this fact, it will be possible to show that $\bar{R}$, the "irreflexive intransitive part" of R, satisfies the conditions of the theorem. To show that R must be antisymmetric, assume $(x_i' + x_j)$ and $(x_j' + x_i)$ both occur in E_R, and consider an injective assignment f in $C(I_=)$. Because of the symmetry between x_i and x_j no generality is lost by letting $f(i) < f(j)$; this implies the existence of a zero-one valued assignment $g \in \theta(f)$ with $g(i) = 0$ and $g(j) = 1$. Since $g \in \theta(f)$, $g \in C(Z_=)$ by Theorem 3.1.3. But E_R evaluates to 0 for g because $x_j' + x_i$ evaluates to 0, a contradiction.

Now let $\bar{R}$ be defined by

$$\bar{R} = \{(i,j) \in R \mid i \neq j \wedge (\forall k)((iRk \wedge kRj) \Rightarrow (k=i \vee k=j))\}$$

$\bar{R}$ is irreflexive and intransitive by construction, and antisymmetric because R is antisymmetric. The only factors in E_R that are not in $E_{\bar{R}}$ are either of the form $(x_i' + x_i)$, which is equivalent to 1, or of the form $(x_i' + x_j)$, with factors of the form $(x_i' + x_{a_1})$, $(x_{a_1}' + x_{a_2})$ ... $(x_{a_r}' + x_j)$ all appearing in $E_{\bar{R}}$ (and E_R). In either case, it is easy to see that omission of these factors from E_R does not change the characteristic function described by E_R, so $E_{\bar{R}}$ and E_R describe the same characteristic function. $\sqcap$

There exist algorithms ([Miller] pp150-175, for example) which reduce an expression E to minimal product of sums (or sum of products) form. Here "minimal" means that any other equivalent expression contains at least as many literals as does the minimal one. Such a minimum expression affords a fairly compact representation for the zero-one valued outputs of a comparator network, especially when the conditions of Theorem 3.2.7 are satisfied.

It is possible to replace any product of sums expression of the form given in Corollary 3.2.5 by an equivalent expression in a form which will be useful in Section 3.4.

<u>Theorem 3.2.9</u> An expression of the form

$$E = \prod_{k=1}^{m} \left[\left(\prod_{i \in A_k} x_i\right)' + \left(\sum_{j \in B_k} x_j\right) \right],$$

with A_k and B_k subsets of D, describes the same characteristic functions as the expression $\hat{E}$ given by

$$\hat{E} = \sum_{R \in \mathcal{R}} \prod_{(p,q) \in R} (x_p' + x_q)$$

where $\mathcal{R}$ is the family of binary relations on D such that $R \in \mathcal{R}$ iff $R \subseteq \left(\bigcup_{k=1}^{m} A_k\right) \times \left(\bigcup_{k=1}^{m} B_k\right)$ and $R \cap (A_k \times B_k)$ is a singleton for $k = 1,2,\ldots m$.

<u>Proof</u>: Rewrite each factor of the form

$$\left(\prod_{i \in A_k} x_i\right)' + \left(\sum_{j \in B_k} x_j\right) = \left(\sum_{i \in A_k} x_i'\right) + \left(\sum_{j \in B_k} x_j\right)$$

in the product, duplicating terms as necessary to obtain a factor of the form

$$\sum_{(i,j) \in A_k \times B_k} (x_i' + x_j)$$

Then multiply all the resulting factors to yield

$$\sum_{R \in \mathcal{R}} \prod_{(p,q) \in R} (x_p' + x_q)$$

where $\mathcal{R}$ is the family of binary relations described in the statement of the theorem. □

<u>Example 3.2.10</u> The expression

$$E = (x_0' + x_1' + x_2)\ (x_2' + x_1 + x_3)\ (x_3' + x_1 + x_4)$$

is of the form

$$E = \prod_{k=1}^{3} \left[\left(\prod_{i \in A_k} x_i \right)' + \left(\sum_{j \in B_k} x_j \right) \right]$$

with $A_1 = \{0,1\}$, $A_2 = \{2\}$, $A_3 = \{3\}$, $B_1 = \{2\}$, $B_2 = \{1,3\}$, and $B_3 = \{1,4\}$. Rewriting the factors of E gives

$$[(x_0'+x_2) + (x_1'+x_2)]\ [(x_2'+x_1) + (x_2'+x_3)]\ [(x_3'+x_1) + (x_3'+x_4)]$$

Multiplying out, we get

$$(x_0'+x_2)(x_2'+x_1)(x_3'+x_1)+\ldots+(x_1'+x_2)(x_2'+x_3)(x_3'+x_4)$$

which is an expression of the form

$$\hat{E} = \sum_{R \in \mathcal{R}} \prod_{(p,q) \in R} x_p' + x_q)$$

where $\mathcal{R}$ is the family of relations

$$\{ \{(0,2),\ (2,1),\ (3,1)\},\ \{(0,2),\ (2,1),\ (3,4)\},$$
$$\{(0,2),\ (2,3),\ (3,1)\},\ \{(0,2),\ (2,3),\ (3,4)\},$$
$$\{(1,2),\ (2,1),\ (3,1)\},\ \{(1,2),\ (2,1),\ (3,4)\},$$
$$\{(1,2),\ (2,3),\ (3,1)\},\ \{(1,2),\ (2,3),\ (3,4)\} \}$$

$\equiv$

By a construction similar to that used in Theorem 3.2.8, the binary relations R in $\mathcal{R}$ can be replaced by irreflexive and intransitive relations $\bar{R}$; this is because the "reflexive and transitive parts" of R are redundant. Such a replacement will result in a reduction in the size of the expression $\hat{E}$. It is not obvious that each R in $\mathcal{R}$ is antisymmetric, however; Theorem 3.2.8 does not apply because the expressions E_R do not individually describe the <u>entire</u> set of output assignments. Moreover, a considerably smaller family of relations could be constructed than that arising from the procedure used in the proof; for example the proof procedure would rewrite $(x_1'+x_2'+x_3+x_4)$ as the redundant expression

$$(x_1'+x_3) + (x_1'+x_4) + (x_2'+x_3) + (x_2'+x_4)$$

rather than the more compact expression

$$(x_1'+x_3) + (x_2'+x_4)$$

These two peculiarities of the theorem are related; Theorem 3.4.17 will demonstrate that the redundancy in the family $\mathcal{R}$ will allow $\mathcal{R}$ to be reduced to a family of covering relations; the members of $\mathcal{R}$ whose reflexive transitive closure is not antisymmetric can be discarded. It will follow from this and other considerations in Section 3.4 that the set of zero-one valued assignments at the output of any comparator network can be characterized by a family of partial orders.

3.3 Injective output assignments

In contrast to the zero-one valued case, it is easily shown that $C(I_=)$, the set of injective assignments that can appear at the output of a comparator network, can be characterized by a set of partial orders.

Theorem 3.3.1 For any comparator network C,

$$C(I_=) = \bigcup_{P \in \mathcal{P}} I_P$$

where $\mathcal{P}$ is a (finite) family of partial orders P.

Proof: Let $\mathcal{P} = \{P \mid (P \text{ is a total order on } D) \wedge (\exists f \in C(I_=))\ (f \in I_P)\}$ Since D is a finite set, $\mathcal{P}$ is finite as well, and certainly

$$C(I_=) \subseteq \bigcup_{P \in \mathcal{P}} I_P$$

Now let f' be an element of I_P for some $P \in \mathcal{P}$, so that for all x and y in D, $xPy \Rightarrow f'(x) \leq f'(y)$. Now P is in $\mathcal{P}$ by virtue of the existence of an assignment f in $C(I_=)$ such that $xPy \Rightarrow f(x) \leq f(y)$. Since f is injective and P is total, $f(x) \leq f(y) \Rightarrow xPy$, so that $f(x) \leq f(y) \Rightarrow f'(x) \leq f'(y)$. Finally, to show $f' \in C(I_=)$, let π be a permutation on D such that $C(\pi \cdot f) = f$; it is easily verified that $C(\pi \cdot f') = f'$ using Theorem 2.1.4. $\sqcap$

Theorem 3.3.3 will show that it is in general unnecessary to require that every member of $\mathcal{P}$ be total. It will be useful to prove the following important theorem first.

Theorem 3.3.2 If P_1 and P_2 are partial orders, then $P_1 \subseteq P_2 \Leftrightarrow I_{P_2} \subseteq I_{P_1}$

Proof: ($\Rightarrow$) if $P_1 \subseteq P_2$, then $xP_1y \Rightarrow xP_2y$ for all $x, y \in D$. By definition of I_{P_2}, if $f \in I_{P_2}$ then $xP_2y \Rightarrow f(x) \leq f(y)$ and therefore $xP_1y \Rightarrow f(x) \leq f(y)$, i.e. $f \in I_{P_1}$. ($\Leftarrow$) Let $P_1 \nsubseteq P_2$, so that there exist elements a and b in D with aP_1b and $\neg(aP_2b)$. Let T be a total order containing P_2 such that bTa, i.e. any total order containing the partial order obtained by taking the transitive closure of the set $P_2 \cup \{(b,a)\}$. If $f \in I_T$ then $xP_2y \Rightarrow xTy \Rightarrow f(x) \leq f(y)$ and $f \in I_{P_2}$, but since bTa, $f(b) \leq f(a)$ and $f \notin I_{P_1}$ because f is injective. $\sqcap$

Theorem 3.3.3 If I is a set of injective assignments satisfying

$$I = \bigcup_{p \in \mathcal{P}} I_p$$

for a family of partial orders $\mathcal{P}$ and if $P' = \bigcap \mathcal{P}$, then $I = I_{p'}$ iff every total order contain P' also contains some $P \in \mathcal{P}$.

Proof: ($\Rightarrow$) If $I = I_{P'}$ and $P' \subseteq T$ for some total order T, then $I_T \subseteq I_{P'}$ by Theorem 3.3.2 and hence $I_T \subseteq I$. If f is any element of I_T, $f \in I$ implies $f \in I_P$ for some P, and for all x and y, $xPy \Rightarrow f(x) \leq f(y)$. Since f is injective and T is total, $f(x) \leq f(y) \Rightarrow xTy$, so $P \subseteq T$.

($\Leftarrow$) Certainly $I \subseteq I_{P'}$ since $P' \subseteq P$ for every $P \in \mathcal{P}$. If $f \in I_{P'}$, then let T be the total order defined by f, i.e. the total order with $f \in I_T$. It is easy to show that $P' \subseteq T$. But by hypothesis,

$P \subseteq T$ for some $P \in \mathcal{P}$. This implies that $f \in I_T \subseteq I_p \subseteq I$ and since f was arbitrary, $I_{P'} = I$. □

Theorem 3.3.3 states that some subsets of the family $\mathcal{P}$ of total orders discussed in Theorem 3.3.1 may be describable by intersections over those subsets, thereby giving a smaller family of partial orders.

It remains to consider the effect of a comparator on a set I of injective assignments.

<u>Theorem 3.3.4</u> For any partial order P the comparator $\langle i,j\rangle$ transforms I_P into the set

$$\hat{I}_P = I_{P_1} \cup I_{P_2}$$

where P_1 is the smallest partial order containing P and (i,j), P_2 is the smallest partial order containing $\pi^{-1}.P.\pi$ and (i,j), and π is the permutation (ij).

<u>Proof</u>: Let $D = \{0,1,\ldots,n-1\}$ and let $I_P = L_P \cup G_P$, with

$$L_P = \{f \in I_P \mid f(i) \leq f(j)\}$$
$$G_P = \{f \in I_P \mid f(i) \geq f(j)\}$$

If jPi, then L_P is empty; otherwise let P_1 be the smallest partial order containing P and (i,j). Certainly $P \subseteq P_1$ so that $I_{P_1} \subseteq I_P$. Since iP_1j, every f in I_{P_1} satisfies $f(i) \leq f(j)$, so $I_{P_1} \subseteq L_P$. Now if $f \in L_P$ then there exists a total order T with $f \in I_T$, $P \subseteq T$, and iTj. Since P_1 is the smallest partial order satisfying $P \subseteq P_1$ and iPj, $P_1 \subseteq T$ and $f \in I_T \subseteq I_{P_1}$. Therefore $L_P = I_{P_1}$. The comparator $<i,j>$ has no effect on any element of L_P, so that the image of L_P under the comparator is $L_P = I_{P_1}$.

If iPj then G_P is empty; otherwise, let P_2 be the smallest order containing $\pi^{-1}.P.\pi$ and (i,j). Certainly $P_2 \subseteq \pi^{-1}.P.\pi$, so that $I_{P_2} \subseteq I_{\pi^{-1}.P.\pi}$. Now $\pi^{-1}.G_P = \{f \mid \pi.f \in G_P\}$ is precisely the same set as $\{f \in I_{\pi^{-1}.P.\pi} \mid f(i) \leq f(j)\}$ because

$$\begin{aligned} f \in \pi^{-1}.G_P &\Leftrightarrow (xPy \Rightarrow f(\pi(x)) \leq f(\pi(y))) \\ &\quad \wedge f(\pi(i)) \geq f(\pi(j)) \\ &\Leftrightarrow (\pi^{-1}(x)P\pi^{-1}(y) \Rightarrow f(x) \leq f(y)) \\ &\quad \wedge f(i) \leq f(j) \end{aligned}$$

Since every element of I_{P_2} satisfies $f(i) \leq f(j)$, $I_{P_2} \subseteq \pi^{-1}.G_P$. Select an $f \in \pi^{-1}.G_P$, and let T be the total order with $f \in I_T$. Now $\pi^{-1}.P.\pi \subseteq I_T$ and iTj; since P_2 is the smallest total order satisfying these conditions, $P_2 \subseteq T$ and $f \in I_T \subseteq I_{P_2}$. Therefore $\pi^{-1}.G_P = I_{P_2}$. The comparator $<i,j>$ transforms every element f in G_P into $\pi^{-1}.f$, so that the image of G_P under $<i,j>$ is I_{P_2}.

Putting the two images together, the comparator transforms I_P to I_{P_1} if iPj, I_{P_2} if jPi, and $I_{P_1} \cup I_{P_2}$ if neither is true. $\square$

<u>Corollary 3.3.5</u> The comparator $<i,j>$ [illegible]sforms the set of injective assignments

$$I = \bigcup_{P \in \mathcal{P}} I_P$$

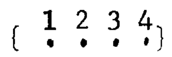
into the set

$$\hat{I} = \bigcup_{P_1 \in \mathcal{P}_1} I_{P_1} \cup \bigcup_{P_2 \in \mathcal{P}_2} I_{P_2}$$

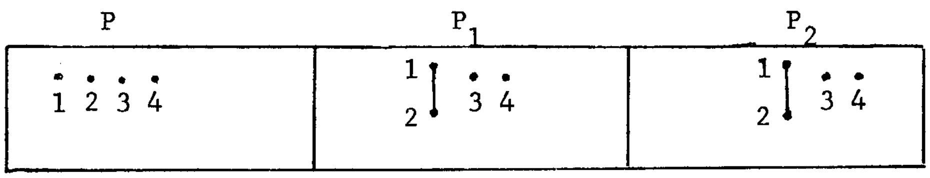
where $\mathcal{P}_1 = \{P_1 \mid (\exists P \in \mathcal{P})(P_1$ is the smallest partial order containing P and $(i,j))\}$, $\mathcal{P}_2 = \{P_2 \mid (\exists P \in \mathcal{P})(P_2$ is the smallest partial order containing $\pi^{-1} \cdot P \cdot \pi$ and $(i,j))\}$, and π is the permutation (ij). □

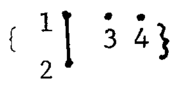
An example is in order.

Example 3.3.6 Consider the comparator network $\langle 1,2\rangle \cdot \langle 3,4\rangle \cdot \langle 1,3\rangle \cdot \langle 2,4\rangle \cdot \langle 2,3\rangle$. The family of partial orders $\mathcal{P}$ with $C(I_=) = \bigcup_{P \in \mathcal{P}} I_P$ is given below for each stage in the network. For each comparator in the network and for each $P \in \mathcal{P}_i$, the partial orders P_1 and P_2 are given, and the next family $\mathcal{P}_{i+1}$ is of course the set consisting of all P_1 and P_2 arising from partial orders P in the preceding family $\mathcal{P}_i$.

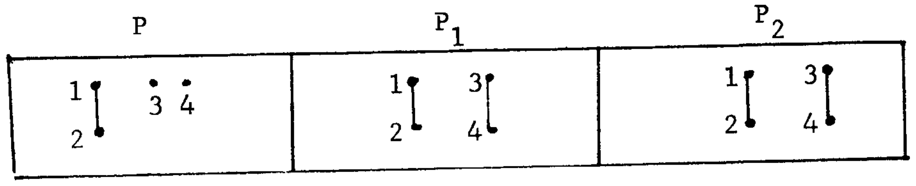
$\mathcal{P}_0 =$ 

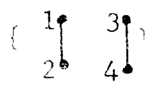
$\langle 1,2\rangle$:



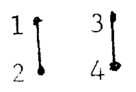
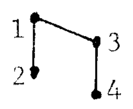
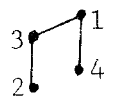
$\mathcal{P}_1 =$ 

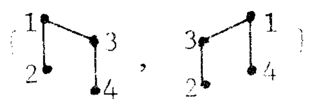
$\langle 3,4\rangle$:



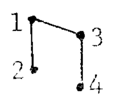
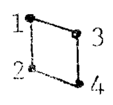
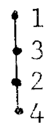
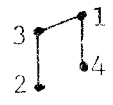
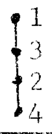
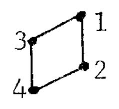
P_2 = 

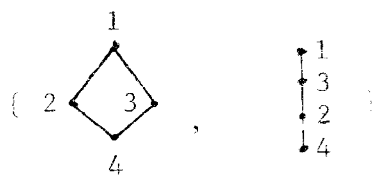
$<1,3>$:

P	P_1	P_2
		

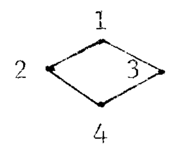
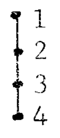
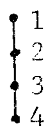
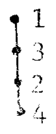
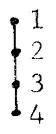
P_3 = 

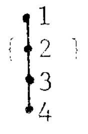
$<2,4>$:

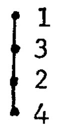
P	P_1	P_2
		
		

P_4 = 

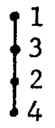
$<2,3>$:

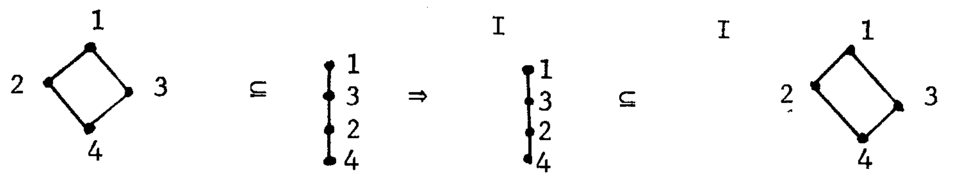
P	P_1	P_2
		
	none	

P_5 = 

Notice that there is no partial order P_1 arising from the partial order  and the comparator <2,3>, and that

$$\bigcup_{P\in\mathcal{P}_3} I_P$$

cannot be expressed as $I_{P'}$ for any partial order P'. Also, the total orders  occurring after comparator <2,4> could have been eliminated via the criterion of Theorem 3.3.3 or merely by observing that



. □

In section 3.4, the notion of a lattice of partial orders on a set will be introduced. This notion will help elucidate the connection between zero-one valued assignments and injective assignments.

3.4 The lattice of partial orders

In sections 3.2 and 3.3, two techniques for characterizing network outputs were developed which have many similar features. In both cases, for example, the effect of a comparator on the output set is reflected in the characterization by the formation of a union, and when certain conditions are met the characterization can be reduced to a single partial order. The lattice of partial orders on a domain is a convenient vehicle for exploring this similarity further.

Definiton 3.4.1 Let 0_D denote the set of all partial orders on the set D together with the relation $D \times D = D^2$. □

The proof of the next theorem is straightforward.

Theorem 3.4.2 The set 0_D, ordered by $\subseteq$, forms a lattice with respect to the operations of set intersection and supremum, where the supremum operation $\vee$ is defined in the usual way:

$$P_1 \vee P_2 = \cap \{ P \in 0_D \mid P_1 \subseteq P \wedge P_2 \leq P \}$$ □

Figure 3.4.1 depicts the lattice $0_{\{0,1,2\}}$. The lattice 0_D is not modular, and therefore not distributive, for domains of more than two elements. However 0_D is useful because of Theorem 3.3.2 and the following result.

Theorem 3.4.3 If P_1 and P_2 are partial orders, then $P_1 \subseteq P_2 \Leftrightarrow Z_{P_2} \subseteq Z_{P_1}$.

Proof: ($\Rightarrow$) Let $P_1 \subseteq P_2$ with $g \in Z_{P_2}$. Then $(\forall x, y \in D)(xP_1y \Rightarrow xP_2y \wedge xP_2y \Rightarrow g(x) \leq g(y))$, so $g \in Z_{P_1}$ and $Z_{P_2} \subseteq Z_{P_1}$. ($\Leftarrow$) Let $Z_{P_2} \subseteq Z_{P_1}$, and suppose x and y are such that xP_1y but $\neg (xP_2y)$. Then let the assignment g be defined by

$$g(\omega) = \begin{cases} 1 & \text{if } xP_2\omega \\ 0 & \text{else} \end{cases}$$

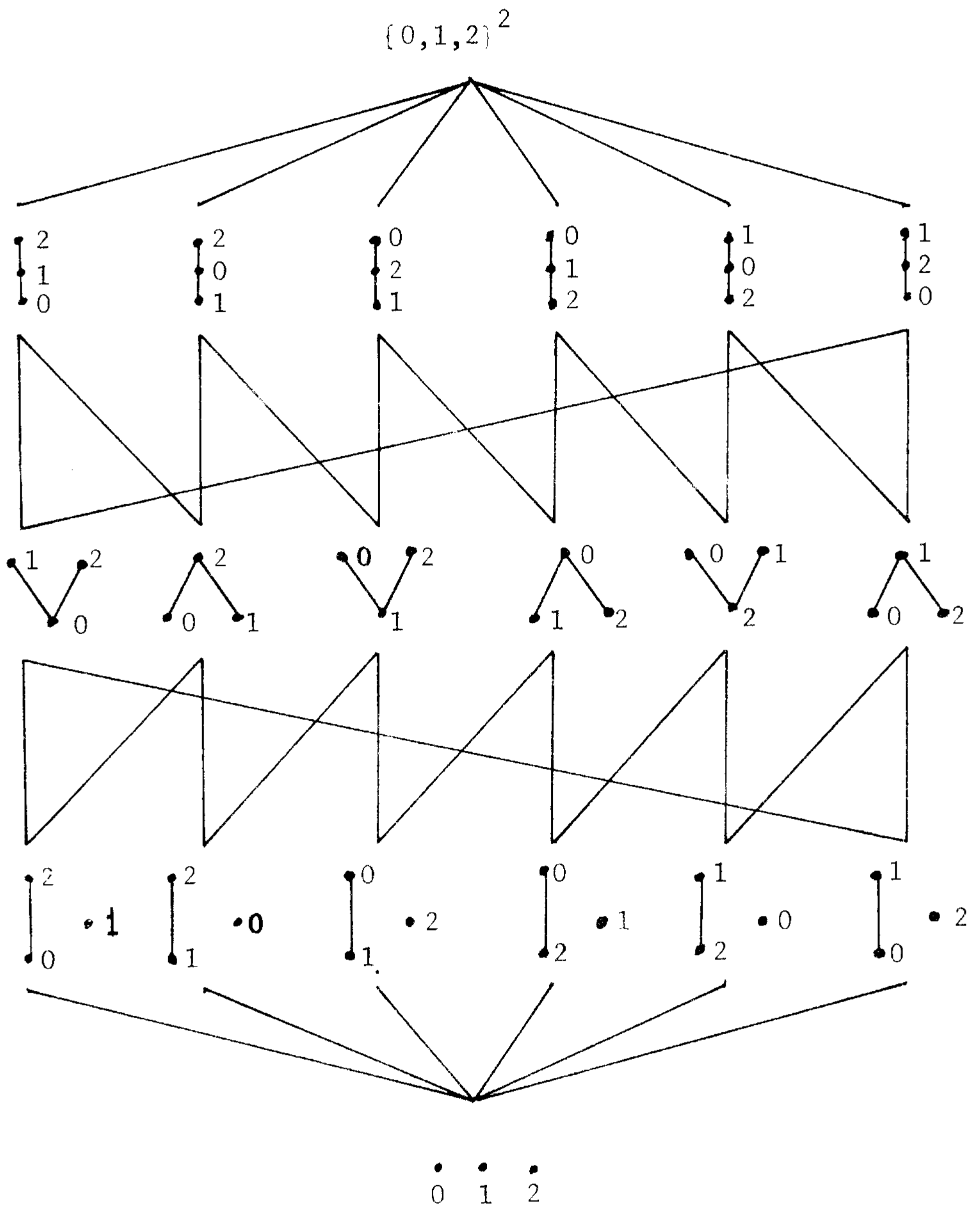


Figure 3.4.1 The lattice $0_{\{0,1,2\}}$

Now $g(x) = 1$ and $g(y) = 0$, so $g \notin Z_{P_1}$. Since $Z_{P_2} \subseteq Z_{P_1}$, $g \notin Z_{P_2}$ implying that for some a and b in D, aP_2b, $g(a) = 1$, and $g(b) = 0$. By definition of g, xP_2a, so by transitivity of P_2, xP_2b and $g(b) = 1$, a contradiction. $\sqcap$

Theorems 3.3.2 and 3.4.3 state that the assignment functions I and Z are monotone decreasing functions from the lattice of partial orders 0_D to the power sets of the injective and zero-one valued assignments respectively. (Setting $I_{D^2} = \emptyset$ and $Z_{D^2} = \{\{0\}^D, \{1\}^D\}$ is consistent not only with the definitions but also with these theorems.) The next theorem follows from Theorems 3.3.2 and 3.4.3.

<u>Theorem 3.4.4</u>

(a) $I_{P_1 \vee P_2} = I_{P_1} \cap I_{P_2}$, and

(b) $Z_{P_1 \vee P_2} = Z_{P_1} \cap Z_{P_2}$ if $P_1 \vee P_2 \neq D^2$

<u>Proof</u>: (a) (injective case) Let P_3 be the set of pairs (x,y) consistent with every assignment in both I_1 and I_2; more precisely, let $P_3 = \{(x,y) \mid (\forall f \in I_{P_1} \cap I_{P_2}) (f(x) \leq f(y))\}$. It is readily verified that P_3 is a partial order. Moreover, $I_{P_1} \cap I_{P_2} \subseteq I_{P_3}$, for if $f \in I_{P_1} \cap I_{P_2}$ and xPy, then $f(x) \leq f(y)$ by definition of P_3. Now $P_1 \subseteq P_3$, for if xP_1y, then by definition of I_{P_1}, $f(x) \leq f(y)$ for every $f \in I_{P_1}$ and hence for every $f \in I_{P_1} \cap I_{P_2}$; similarly,

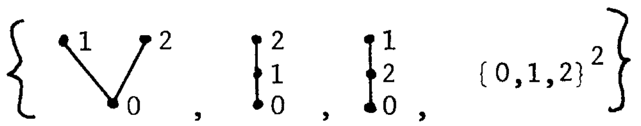
$P_2 \subseteq P_3$. It follows that $I_{P_3} \subseteq I_{P_1}$ and $I_{P_3} \subseteq I_{P_2}$, so that $I_{P_3} \subseteq I_{P_1} \cap I_{P_2}$ and $I_{P_3} = I_{P_1} \cap I_{P_2}$. It remains only to show that $P_3 = P_1 \vee P_2$. Since $P_1 \subseteq P_3$ and $P_2 \subseteq P_3$ have already been proved, let P_4 be such that $P_1 \subseteq P_4$ and $P_2 \subseteq P_4$. Then $I_{P_4} \subseteq I_{P_1}$ and $I_{P_4} \subseteq I_{P_2}$, whence $I_{P_4} \subseteq I_{P_1} \cap I_{P_2} = I_{P_3}$ and $P_3 \subseteq P_4$.

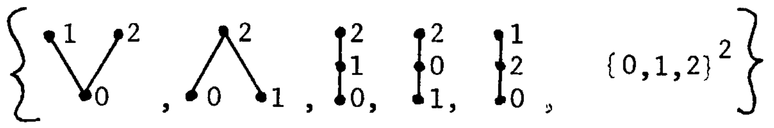
(b) (zero-one case) Let $P_3 = \{(x,y) \mid (\forall f \in Z_{P_1} \cap Z_{P_2})(f(x) \leq f(y))\}$
Now P_3 is certainly reflexive and transitive, with $Z_{P_1} \cap Z_{P_2} \subseteq Z_{P_3}$.
If P_3 is not antisymmetric then for some x and y in D, $x \neq y$ but $f(x) = f(y)$ for all f in $Z_{P_1} \cap Z_{P_2}$. This can only happen if xP_1y and yP_2x or yP_1x and xP_2y; in either event, $P_1 \vee P_2 = D^2$. The remainder of the proof parallels the proof for the injective case. □

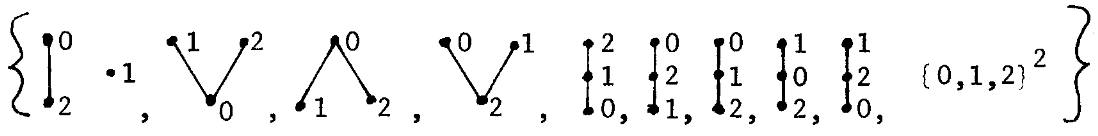
Theorem 3.4.4 suggests that sets of partial orders exhibiting closure properties under supremum may be useful.

<u>Definition 3.4.5</u> A subset F of 0_D is a <u>filter</u> of 0_D if, for all $P \in F$ and all $Q \in 0_D$, $P \vee Q \in F$. Alternatively, F is a filter if, whenever P is in F and $P \subseteq Q$, then Q is in F. □

<u>Example 3.4.6</u> The sets







are all filters of the lattice $0_{\{0,1,2\}}$ shown in Figure 3.4.1 $\square$

<u>Definition 3.4.7</u> If I is a set of injective assignments, $F(I) = \{P \in 0_D \mid I_P \subseteq I\}$; if Z is a set of zero-one valued assignments, $F(Z) = \{P \in 0_D \mid Z_P \subseteq Z\}$. $\square$

<u>Theorem 3.4.8</u> For any set of injective assignments I, F(I) is a filter. Similarly, for any set of zero-one valued assignments Z, F(Z) is a filter.

<u>Proof</u>: Let $P \in F(I)$ and $Q \in 0_D$. Since $P \subseteq P \vee Q$, $I_{P \vee Q} \subseteq I_P \subseteq I$ so $P \vee Q \in F(I)$. The proof for Z is similar. $\square$

The set of generators of a filter is merely its set of minimal elements.

<u>Definition 3.4.9</u> The set of generators of a filter F is the set

$$G(F) = \{P \in F \mid (\forall Q \in F)(Q \subseteq P \Rightarrow Q = P)\}.$$

If G(F) is a singleton {P}, then F is said to be <u>principal</u>. The principal filter generated by P is written (P). $\square$

The connection will now be made between filters and sets of output assignments. It will be shown that for any comparator network C on D,

$$C(I_{=}) = \bigcup_{P \in F(C(I_{=}))} I_P$$

$$C(Z_{=}) = \bigcup_{P \in F(C(Z_{=}))} Z_P$$

The injective case is easily resolved; Theorem 3.3.1 guarantees that

$$C(I_{=}) = \bigcup_{P \in \mathcal{P}} I_P$$

for some family of partial orders $\mathcal{P}$. Certainly $\mathcal{P} \subseteq F(C(I_{=})$, and since $P \in F(C(I_{=})) \Rightarrow I_P \subseteq C(I_{=})$,

$$C(I_{=}) = \bigcup_{P \in F(C(I_{=}))} I_P$$

The situation for zero-one valued assignments is not so simple. First note that the set of zero-one valued assignments $C(Z_{=})$ that can appear at the output of a comparator network C can be described in terms of $C(I_{=})$. This is by virtue of Theorem 3.1.3 and because the set of all zero-one valued assignments on D is just the set of thresholds of the set of all injective assignments on D. Hence

$$C(Z_{=}) = \theta(C(I_{=})) = \bigcup_{P \in \mathcal{P}} \theta(I_P)$$

for some family of partial orders $\mathcal{P}$. The next theorem relates I_P and Z_P.

<u>Theorem 3.4.10</u> If P is a partial order, $\theta(I_P) = Z_P$.

<u>Proof</u>: Let $g \in \theta(I_P)$, so that $g \in \theta(f)$ for some $f \in I_P$. If $f(x) \leq f(y)$ then $g(x) \leq g(y)$; since $f \in I_P$, $xPy \Rightarrow f(x) \leq f(y)$ and $g \in Z_P$. Conversely, let $g \in Z_P$, and without loss of generality let the range of the assignments in I_P be the integers. Certainly I_P contains some assignment f with range $\{0,1,2,\ldots n-1\}$ (where D has n elements) so let h be the assignment defined by $h(x) = f(x) + n.g(x)$. Now h is injective, for if $h(x) = h(y)$ then $h(x) \equiv h(y) \pmod n$ so $f(x) = f(y)$, implying $x = y$. Moreover, h is an element of I_P, since $xPy \Rightarrow f(x) \leq f(y) \wedge g(x) \leq g(y)$. Since $g(x) = 1$ iff $h(x) \geq n$, $g \in \theta(h) \subseteq \theta(I_P)$. Therefore $g \in \theta\ (I_P) \Leftrightarrow g \in Z_P$, so $\theta(I_P) = Z_P$. □

Theorem 3.4.10 and the remarks preceding it imply that

$$C(Z_=) = \bigcup_{P \in \mathcal{P}} Z_P$$

Since $\mathcal{P} \subseteq F(C(Z_=))$ obviously and $P \in F(C(Z_=)) \Rightarrow Z_P \subseteq C(Z_=)$,

$$C(Z_=) = \bigcup_{P \in F(C(Z_=))} Z_P$$

The foregoing discussion is recapitulated in the following theorem.

<u>Theorem 3.4.11</u> For any comparator network C,

$$C(I_=) = \bigcup_{P \in F(C(I_=))} I_P$$

and

$$C(Z_=) = \theta(C(I_=)) = \bigcup_{P \in F(C(Z_=))} Z_P$$ □

The next theorem deals with the cases $I = I_P$ and $Z = Z_P$.

Theorem 3.4.12 $F(I_P) = F(Z_P) = (P)$

Proof: Let $Q \in F(I_P)$; then $I_Q \subseteq I_P$ and $P \subseteq Q$, so $Q \in (P)$. If $Q \in (P)$, then $P \subseteq Q$, $I_Q \subseteq I_P$, and $Q \in F(I_P)$. The proof for Z_P is entirely similar. $\square$

As an immediate consequence of Theorems 3.4.11 and 3.4.12, we have

Corollary 3.4.13. For any partial order P,

$$F(C(I_=)) = (P) \Leftrightarrow C(I_=) = I_P$$
$$F(C(Z_=)) = (P) \Leftrightarrow C(Z_=) = Z_P. \qquad \square$$

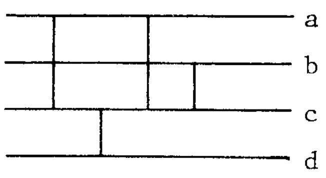
Theorems 3.4.10 and 3.4.11 and Corollary 3.4.13 provide the raw material for Theorem 3.4.14.

Theorem 3.4.14. For any comparator network C, if $F(C(I_=)) = (P)$ for some partial order P, then $F(C(Z_=)) = (P)$ as well.

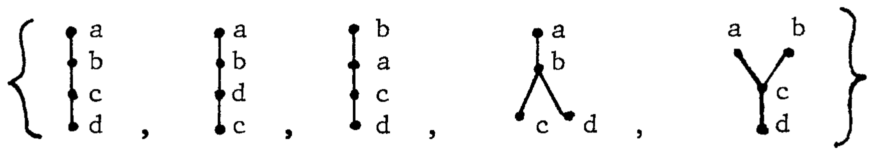
Proof: If $F(C(I_=)) = (P)$, then $C(I_=) = I_P$ and $C(Z_=) = \theta(C(I_=)) = \theta(I_P) = Z_P$, so that $F(C(Z_=)) = (P)$. $\square$

The converse of this theorem is false, and the example below demonstrates.

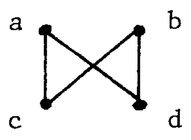
Example 3.4.15. For the network of Example 3.1.6, namely



the corresponding filter $F(C(I_=))$ is



This filter is not principal; it is generated by the last two elements listed. $F(\theta(C(I_=)))$, on the other hand, is principal and is generated by



$\square$

In general, then, $F(C(I_=))$ and $F(\theta(C(I_=)))$ are different filters with different generating sets. They are not totally unrelated; the filter $F(\theta(C(I_=)))$ always contains the filter $F(C(I_=))$.

<u>Theorem 3.4.16</u> $F(I) \subseteq F(\theta(I))$ for any set of injective assignments I.

<u>Proof</u>: If $P \in F(I)$ then $I_P \subseteq I$, implying $Z_P = \theta(I_P) \subseteq \theta(I)$ and $P \in F(\theta(I))$. $\square$

When $F(C(I_=))$ is principal, this fact can be determined by applying Theorem 3.3.3 to any family of partial orders $\mathcal{P}$ such that

$$C(I_=) = \bigcup_{P\in\mathcal{P}} I_P$$

If $F(C(\underline{\underline{I}}))$ is not principal, it is not clear how to obtain the generators of the filter without first constructing the entire filter. The generators of $F(C(\underline{\underline{Z}}))$ may be obtained in an easier fashion from any product of sums expression for Z.

<u>Theorem 3.4.17</u> Let E be a product of sums expressions for the characteristic function of $C(\underline{\underline{Z}})$ in the form specified by Corollary 3.2.5, so that

$$E = \prod_{k=1}^{m} \left[\left(\prod_{i \in A_k} x_i \right)' + \left(\sum_{j \in B_k} x_j \right) \right]$$

Let $\hat{E}$ be the equivalent expression defined in Theorem 3.2.9, so that

$$\hat{E} = \sum_{R \in \mathcal{R}} \prod_{(p,q) \in R} (x_p' + x_q)$$

where $\mathcal{R}$ is a family of binary relations R with $R \cap (A_k \times B_k)$ a singleton for $k = 1,2,\ldots m$. Then every member of $F(C(\underline{\underline{Z}})$ contains some $R \in \mathcal{R}$.

<u>Proof</u>: Let P be a member of $F(C(\underline{\underline{Z}}))$, so that $Z_P \subseteq C(\underline{\underline{Z}})$. Then any expression describing the characteristic function for Z_P, and in particular the expression E_P given by

$$E_P = \prod_{(p,q) \in P} (x_p' + x_q)$$

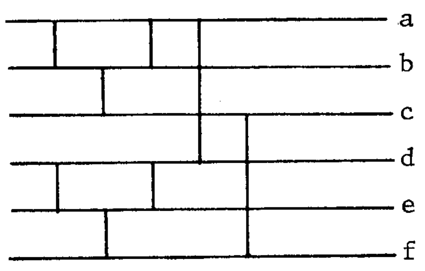
implies $\hat{E}$; that is, if E_P evaluates to 1, so does $\hat{E}$. It will be shown that $P \cap (A_k \times B_k)$ is nonempty for $k = 1,2,\ldots m$, implying $R \subseteq P$ for some $R \in \mathcal{R}$. Suppose $P \cap (A_k \times B_k)$ is empty for some k. Let g be the assignment

$$g(\omega) \begin{cases} 1 \text{ if } (\exists a \in A_k)\,(aP\omega) \\ 0 \text{ else} \end{cases}$$

First, $g \in Z_P$; if this were not so, there would exist x and y such that xPy, $g(x) = 1$, and $g(y) = 0$. But if $g(x) = 1$, aPx for some $a \in A_k$; by transitivity of P, aPy and $g(y) = 1$, a contradiction. $g(a) = 1$ for every $a \in A_k$ by definition of g, but $g(b) = 0$ for every $b \in B_k$; if this were not so, then $P \cap A_k \times B_k$ would be nonempty. Now E_P evaluated at g is clearly zero, since its kth term is zero, so $\hat{E}$ evaluated at g is zero, a contradiction of the fact that E_P implies $\hat{E}$. Hence $R \subseteq P$ for some $R \in \mathcal{R}$. $\square$

The statement of the theorem is much more cumbersome than its application in practice.

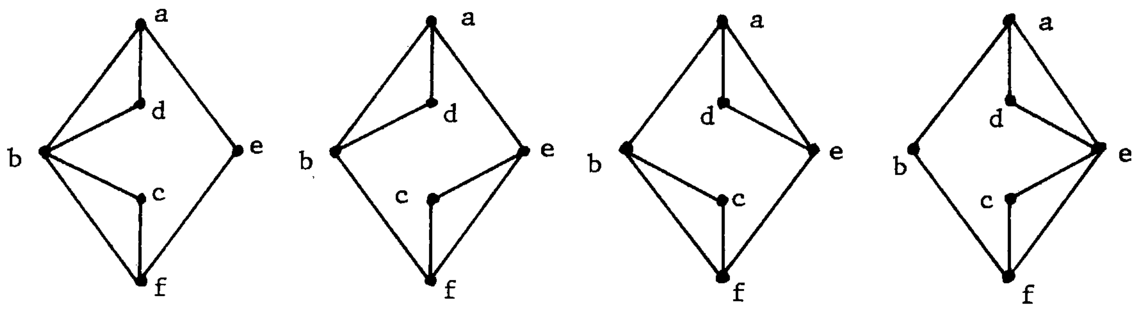
<u>Example 3.4.18</u> Consider the network shown below.



One expression for the characteristic function of the zero-one valued outputs $C(Z_=)$ is

$$E = (x_a'+x_b)(x_a'+x_d)(x_a'+x_e)(x_d'+x_b+x_e)$$
$$(x_b'+x_e'+x_c)(x_b'+x_f)(x_c'+x_f)(x_e'+x_f)$$

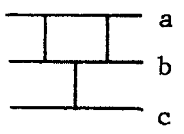
According to Theorem 3.4.16, any partial order in $F(C(Z_=))$ must contain (a,b), (a,d),(a,e),(b,f),(c,f),(e,f) and one element from each of the sets {(d,b),(d,e)} and {(b,c),(e,c)}. That is, any partial order in $F(C(Z_=))$ must contain one of the four partial orders



Since none of these partial orders is contained in any other, they are the generators of $F(C(Z_=))$. □

Note that if there is no partial order containing a relation $R \in \mathcal{R}$ then that R may be discarded for purposes of finding the generators of $F(C(Z_=))$. This will occur if the expressions for $C(Z_=)$ gives rise to relations R whose transitive closure is not antisymmetric. An example is the expression

$$E = (x_a'+x_b)(x_b'+x_a+x_c)(x_a'+x_c)$$

which describes the outputs of the sorting network  albeit not minimally. It also may happen that for some R and R' in the smallest partial order containing R also contains R'; in this event R may be discarded.

The results of this chapter indicate two things: first, that zero-one valued assignments are usually more convenient than injective assignments as a tool for characterizing output behavior; and second, that sets of assignments described by single partial orders are particularly simple to work with. Chapter 4 explores further the properties of sets of assignments described by single partial orders.

CHAPTER 4

OUTPUTS CHARACTERIZED BY A PARTIAL ORDER

4.1 Sorting Injective Assignments with respect to a Partial Order

It is easy to show that for any partial order P on a finite domain there exists a comparator network such that all possible injective outputs of the network are sorted with respect to P; sorting with respect to any total order containing P will suffice. Perhaps not so obvious is the fact that for any partial order P there exists a comparator network such that $C(I_{=})$, the set of possible injective outputs of the network, consists of exactly those assignments that are sorted with respect to P. A lemma will be useful for establishing this result.

Lemma 4.1.1 Let P be a partial order on D with x and y elements of D such that $\neg$ (xPy) and $\neg$ (yPx). Let the set X be

$$X = \{a \in D \mid aPx \wedge \neg (aPy)\}$$

Now, X is nonempty since $x \in X$, so let α be a P-least element of X. Next let the set Y be defined by

$$Y = \{b \in D \mid yPb \wedge \neg (\alpha Pb)\}$$

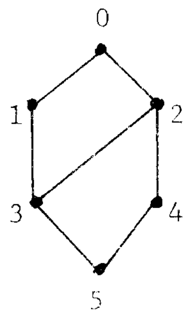
Y is nonempty because $y \in Y$, so let β be a P-greatest element of Y. Then P_1, the smallest partial order containing P and the ordered pair (α,β), is contained in both

P_2, the smallest partial order containing $\pi^{-1}\cdot P\cdot\pi$ and (α,β),

where π is the permutation (α,β), and

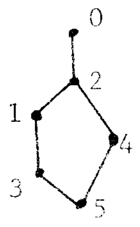
P_3, the smallest partial order containing P and (x,y).

For example, if P is the partial order

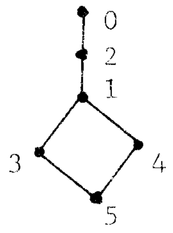


with $x = 4$ and $y = 1$, then $\alpha = 2$, $\beta = 1$,

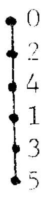
P_1 is



P_2 is



and P_3 is



(remember the "higher is smaller" convention for partial orders.)

<u>Proof</u>: To show $P_1 \subseteq P_2$ it suffices to show $P \subseteq P_2$. Let aPb.

Case 1. $a = \alpha$. Now $b \neq \beta$, because $\alpha P \beta$ contradicts $\beta \in Y$. Therefore $\beta P_2 b$, and since $\alpha P_2 \beta$, $\alpha P_2 b$, i.e. $aP_2 b$.

Case 2. $a = \beta$. Now $b \neq \alpha$, because if $\beta P \alpha$, then αPx and $yP\beta$ would imply yPx. Since β is a greatest element of Y, either $b = \beta$ or αPb. In either event $\beta P_2 b$, i.e. $aP_2 b$.

Case 3. $a \neq \alpha$, $a \neq \beta$, $b = \alpha$. Then since α is a least element of X, aPy; and since $yP\beta$, $aP\beta$ and $aP_2\alpha$, i.e. $aP_2 b$.

Case 4. $a \neq \alpha$, $a \neq \beta$, $b = \beta$. Then $aP_2\alpha$ and since $\alpha P_2 \beta$, $aP_2\beta$, i.e. $aP_2 b$.

Case 5. $a \neq \alpha$, $a \neq \beta$, $b \neq \alpha$, $b \neq \beta$. Then aPb implies $aP_2 b$.

To show $P_1 \subseteq P_3$, it suffices to show that $\alpha P_3 \beta$. But since αPx and $yP\beta$, $\alpha P_3 x$ and $yP_3\beta$; since $xP_3 y$, $\alpha P_3 \beta$.

Note that $\alpha P_1 \beta$ but $\neg(\alpha P \beta)$; in fact, it is not difficult to show that $P_1 = P \cup \{(\alpha,\beta)\}$, so that P_1 contains exactly one more ordered pair than does P. □

From this construction it is easy to see how to obtain any set of injective assignments of the form I_P as the set of outputs $C(I_=)$ of some comparator network; in fact, it is possible to construct a network that transforms any I_P into I_Q, as long as I_P contains I_Q.

Theorem 4.1.2 If P and Q are partial orders with $P \subseteq Q$ then there exists a comparator network which transforms I_P into I_Q.

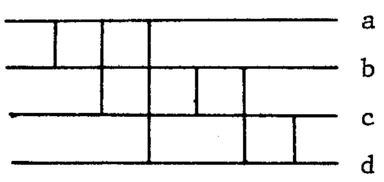
Proof: The proof is by induction on the number of ordered pairs in Q-P.

Basis: If Q-P is empty, then $Q = P$, $I_Q = I_P$, and a vacuous network suffices.

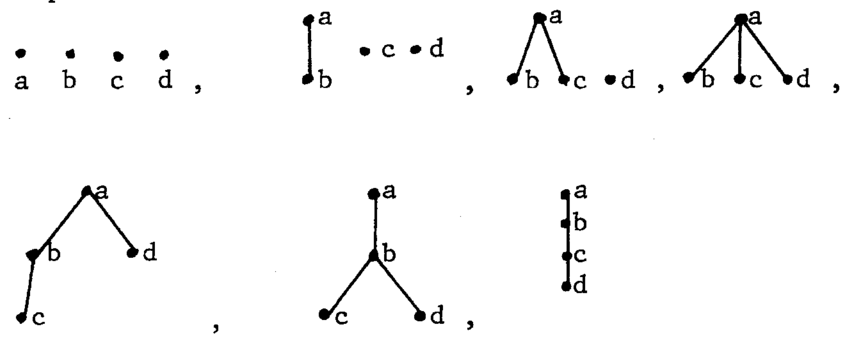
Induction Step: Suppose a comparator network exists to transform I_P to I_Q as long as Q-P contains n or fewer ordered pairs, and suppose Q-P contains n+1 ordered pairs, one of which is (x,y). Apply the comparator $\langle\alpha,\beta\rangle$ determined by Lemma 4.1.1 to I_P; by Theorem 3.3.4, this comparator transforms I_P into $I_{P_1} \cup I_{P_2}$, where P_1 is the smallest partial order containing P and (α,β), and P_2 is the smallest partial order containing $(\alpha\beta)\cdot P\cdot(\alpha\beta)$ and (α,β). By Lemma 4.1.1, $P_1 \subseteq P_2$, so $I_{P_2} \subseteq I_{P_1}$ and I_P is transformed by $\langle\alpha,\beta\rangle$ into I_{P_1}. Since (x,y) is a member of Q and $P \subseteq Q$, Q contains P_3, the smallest partial order containing P and (x,y). Since Lemma 4.1.1 also guarantees $P_1 \subseteq P_3$, $P_1 \subseteq Q$. Now since $\alpha P_1 \beta$ but $\neg(\alpha P \beta)$, $Q-P_1$ has n or fewer elements, so that by the induction hypothesis, I_{P_1} can be transformed to I_Q by some comparator network. It follows that $\langle\alpha,\beta\rangle$ concatenated with this network transforms I_P into I_Q. □

Repeated application of the induction step in the theorem will give rise to a comparator network that transforms I_P into I_Q and that contains as many comparators as there are ordered pairs in Q-P. In particular, if P is the identity relation on D and Q is a total order, the resulting network will be a sorting network containing $\binom{n}{2}$ comparators. Such a network is inefficient, but it has the interesting property that at any stage, the injective assignments that can appear (and, because of Theorem 3.4.14, the zero-one valued assignments that can appear) are characterized by a single partial order.

Example 4.1.3 The sorting network



conforms to the inductive construction of Theorem 4.1.2 for the sequence of partial orders



Lemma 4.1.1 and Theorem 4.1.2 are not the only ways to guarantee a single partial order at each stage; a lemma dual to lemma 4.1.1 can be established which guarantees that the partial orders P_1 and P_2 described in Theorem 3.3.4 satisfy $P_2 \subseteq P_1$. An inductive argument similar to

Theorem 4.1.2 but employing either lemma in the induction step would give rise to a broader class of networks than those derivable from Theorem 4.1.2.

It is sometimes possible to transform I_P into I_Q by sorting I_P with respect to a partial order S.

Theorem 4.1.4 Let P and S be partial orders on D, and let C be a comparator network so that

1. $C(I_=) = I_S$
2. $(\forall f \in I_S)\ (C(f) = f)$
3. $C(I_P) \subseteq I_P$

Then $C(I_P) = I_P \cap I_S = I_{P \vee S}$.

Proof: By statement 3, C maps I_P into a subset of I_P; by statement 1, C maps I_P into a subset of I_S. On the other hand, C maps $I_P \cap I_S$ onto $I_P \cap I_S$ by statement 2. Hence C maps I_P onto $I_P \cap I_S$, which is equal to $I_{P \vee S}$ by Theorem 3.4.4. □

[Gale and Karp] have developed a necessary and sufficient condition for statement 3 to hold when S is a partial order whose maximal chains are disjoint and C is a standard form network that sorts according to the maximal chains of S. Before stating Gale and Karp's result, a definition will be useful.

Definition 4.1.5 Let P be a partial order on domain D. For any element x of D, the sets xP and Px are defined as follows:

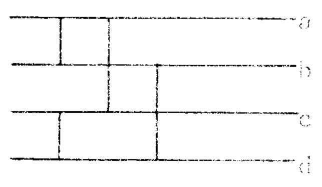
$$xP = \{y \in D \mid xPy\}$$
$$Px = \{y \in D \mid yPx\}$$ □

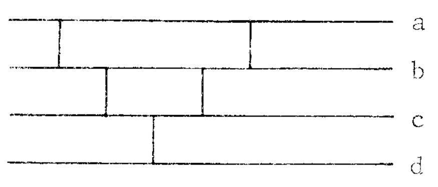
Theorem 4.1.6 [Gale and Karp] Let C be a standard form comparator network such that $C(I_=) = I_S$, where S is a partial order on D with maximal chains disjoint. Further, let P be a partial order on D. Then C transforms every element of I_P into an element of I_P iff

1. $S \cup P$ is antisymmetric;
2. If xPy then for every set $B \subseteq yS \cup Sy$ with $|B| = |Sy|$ there exists a set $A \subseteq xS \cup Sx$ with $|A| = |Sx|$ such that every element of A is P-less than some element of B. $\square$

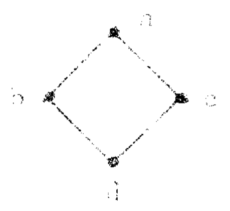
[Liu] has extended this result by showing that if C is a comparator network satisfying the hypothesis of the theorem and if condition 1 of the theorem is satisfied, then $C(I_P)$ is a set I such that Q, the largest partial order satisfying $I \subseteq I_Q$, is equal to $S \vee P'$, where P' is the partial order consisting of all those ordered pairs (x,y) from P that satisfy condition 2 of the theorem. It is readily shown that the largest partial order Q such that $I \subseteq I_Q$ is just $\cap(F(I))$, the intersection of all of the partial orders in the filter F(I). The key fact that makes this construction possible is the fact that S has disjoint maximal chains; if this were not the case, then there might exist two networks which have a different effect on I_P even though both networks sort with respect to the same partial order.

Example 4.1.7   The two networks

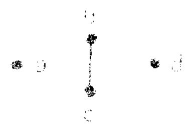




both sort with respect to the partial order S with ordering diagram



Let P be the partial order with ordering diagram



The left-hand network transforms I_P into I_S, whereas the right-hand network sorts every element of I_P. □

In view of the above example, it is probably not possible to generalize the results of Gale and Karp or Liu to deal with the case where S does not have maximal chains disjoint without specifying more about the nature of the comparator network C that sorts with respect to S.

4.2 Real-valued Assignments and Convexity

In this section, we will consider assignments whose range is the set of real numbers. It will be appropriate to refer to such assignments as vectors in real n-dimensional coordinate space R^n. One reason this is convenient is that if a comparator $<i,j>$ on the domain $D = \{0,1,\ldots n-1\}$ is applied to a vector in R^n, then the effect of the comparator can be described by means of a hyperplane in R^n.

Definition 4.2.1. For any two distinct elements i and j in $\{1,2,\ldots n\}$, let H_{ij} be defined by

$$H_{ij} = \{v \in R^n | v_j - v_i = 0\} \qquad \square$$

H_{ij} is certainly a hyperplane (i.e., an n-1 dimensional subspace) of R^n. By the projection theorem, any element u of R^n can be written uniquely as a sum $u = v + w$, where v is an element of the hyperplane H_{ij} and w is an element of $H_{ij}^{\perp}$. (Recall that if H is a subspace of vector space V, $H^{\perp}$ is the set of vectors of V such that the inner product of any element of H and any element of H is zero). From the definition of H_{ij}, $H_{ij}^{\perp}$ is the subspace

$$H_{ij}^{\perp} = \{w \in R^n | w_i + w_j = 0 \wedge (\forall k)$$
$$(k = i \vee k = j \vee w_k = 0\}$$

Now, an element u in R^n lies on the positive side of H_{ij} $(u_j - u_i \geq 0)$ or on the negative side of $H_{ij}(u_j - u_i \leq 0)$ or possibly both; the side of the hyperplane on which u resides is determined by the sign of

u_j-u_i. Notice also that if u_j-u_i is positive, the comparator, $<i,j>$ has no effect on u, whereas if u_j-u_i is negative, the comparator $<i,j>$ interchanges u_i and u_j. If u is written $u = v+w$ with $v \in H_{ij}$ and $w \in H_{ij}^{\perp}$, then clearly $v_i = v_j = \frac{u_i+u_j}{2}$ and

$$w_i = \frac{u_i-u_j}{2} \quad , \quad w_j = \frac{u_j-u_i}{2}$$

If u lies on the positive side of H_{ij}, the comparator transforms $u = v+w$ into $u' = v+w$; if u lies on the negative side of H_{ij}, the comparator transforms $u = v+w$ into $u' = v-w$. Theorem 4.2.2 sums up these observations.

<u>Theorem 4.2.2</u>. If $<i,j>$ is a comparator on the domain $\{1,2,\ldots n\}$ and u is any vector in R^n with $u = v+w$, $v \in H_{ij}$, and $w \in H_{ij}^{\perp}$, then $<i,j>$ transforms u into

$$u' = v + (\text{sign } w_j)w \qquad \square$$

The transformation performed by a comparator is nonlinear. In essence, the comparator $<i,j>$ "reflects" every vector u to the positive side of H_{ij}. It would seem plausible that a comparator network would at least transform the set R^n into a convex set; the remainder of this section will consist of a proof that this is true if and only if the comparator network maps R^n onto the set of vectors consistent with some partial order.

Theorem 4.2.3 The set A of real vectors that can appear at the output of a comparator network C is convex iff $A = A_P$ for some partial order P.

Proof: ($\Leftarrow$) if $A = A_P$ for some partial order P, i.e. if

$$A = \{u \in R^n \mid (\forall i,j)(iPj \Rightarrow u_i \leq u_j)\}$$

then for any two elements u and u' in A and for any λ, $0 \leq \lambda \leq 1$, the vector $\lambda u + (1-\lambda)\, u'$ is in A, since if iPj, then $u_i \leq u_j$ and $u'_i \leq u'_j$; since both λ and $1-\lambda$ are positive, $\lambda u_i \leq \lambda u_j$ and $(1-\lambda)u'_i \leq (1-\lambda)u'_j$. This means that $\lambda u_i + (1-\lambda)u'_i \leq \lambda u_j + (1-\lambda)u'_j$, so $\lambda u + (1-\lambda)u'$ is an element of A.

($\Rightarrow$) Let A be a convex set, and let P be the set $\{(i,j) \mid (\forall u \in A)(u_j - u_i \geq 0)\}$. P is obviously reflexive; P is antisymmetric because A contains injective assignments. P is transitive because if $u_j - u_i \geq 0$ and $u_k - u_j \geq 0$, then $u_k - u_i \geq 0$. Thus P is a partial order. Now $A \subseteq A_P$, since iPj implies $u_i \leq u_j$ for every $u \in A$, so it remains to show that $A_P = A$.

Now A_P and A are surely nonempty since A is the set of possible vectors at the output of C. In fact, A contains injective assignments; according to Theorem 3.3.1, the set I of injective assignments of A can be written

$$I = \bigcup_{T \in \mathcal{T}} I_T$$

where each T is a total order. It is also true that I_P, the set of injective assignments in A_P, can be written

$$I_P = \bigcup_{P \subseteq T} I_T$$

and since $A \subseteq A_P$, $I \subseteq I_P$. Moreover, since $I_T \subseteq I$ for all $T \in \mathcal{T}$, $I_T \subseteq I_P$ and $P \subseteq T$ for every $T \in \mathcal{T}$. It will be shown that $P \subseteq T \Rightarrow T \in \mathcal{T}$; this will imply $I_P = I$, $Z_P = Z$, and finally $A_P = A$.

Let $P \subseteq T$ with $T \notin \mathcal{T}$, and let u be an assignment vector in $I_T \subseteq I_P - I$ with every component u_i a distinct power of 2. In particular let the j^{th} smallest component of u be 2^j, $j = 1,2,\ldots n$. Select any total order $T' \in \mathcal{T}$, and let v be an assignment vector in $I_{T'} \subseteq I$ with every component v_i a distinct power of 2^n, so that the j^{th} smallest component of v is 2^{jn}, $j = 1,2,\ldots n$.

By construction, $u \in A_P - A$ and $v \in A$; since A_P is convex, every vector $w = \lambda u + (1-\lambda)v$ belongs to A_P for all λ in the interval $0 \le \lambda \le 1$, but w belongs to A only for certain values of λ. Since A is convex, the collection of these λ forms an interval $0 \le \lambda < \lambda_0$ or $0 \le \lambda \le \lambda_0$ for some λ_0. $x = \lambda_0 u + (1-\lambda_0)v$ is not injective; if it were, then letting T denote the total order such that $x \in I_T$, every vector within some small neighborhood of x also would belong to I_T. Since this small neighborhood would include both vectors in I and vectors in $I_P - I$ by definition of λ_0, I_T would be a subset of both I and $I_P - I$, a contradiction. Therefore $x_i = x_j$ for some pair i,j with $i \neq j$. It will be argued that this cannot occur for more than one such pair.

Suppose $x_i = x_j$ and $x_k = x_\ell$, with $i \neq j$, $k \neq \ell$, and $\{i,j\} \neq \{k,\ell\}$. Then $\lambda_0 u_i + (1-\lambda_0) v_i = \lambda_0 u_j + (1-\lambda_0)v_j$ and $\lambda_0 u_k + (1-\lambda_0)v_k = \lambda_0 u_\ell + (1-\lambda_0)v_\ell$. Solving these equations for λ_0 yields

$$\lambda_0 = \frac{(v_j - v_i)}{(v_j - v_i) - (u_j - u_i)} = \frac{(v_\ell - v_k)}{(v_\ell - v_k) - (u_\ell - u_k)}$$

or

$$\frac{1}{\lambda_0} = 1 - \frac{(u_j - u_i)}{(v_j - v_i)} = 1 - \frac{(u_\ell - u_k)}{(v_\ell - v_k)}$$

whence

$$\frac{(u_j - u_i)}{(u_\ell - u_k)} = \frac{(v_j - v_i)}{(v_\ell - v_k)}$$

Now either $\left| \frac{(u_j - u_i)}{(u_\ell - u_k)} \right| < 1$, so that

$$\frac{2^2 - 2}{2^n - 2} \leq \left| \frac{(u_j - u_i)}{(u_\ell - u_k)} \right| \leq \frac{2^n - 2^2}{2^n - 2} < 1$$

or $\left| \frac{(u_j - u_i)}{(u_\ell - u_k)} \right| > 1$, implying

$$1 < \frac{2^n - 2}{2^n - 2^2} \leq \left| \frac{(u_j - u_i)}{(u_\ell - u_k)} \right| \leq \frac{2^n - 2}{2^2 - 2}$$

it remains to be shown that $\left| \frac{(v_j - v_i)}{(v_\ell - v_k)} \right|$ lies outside these intervals.

First, consider the case in which max (v_j, v_i) = max (v_ℓ, v_k). In this case, it can be shown that

$$\frac{2^{n^2}-2^{n^2-n}}{2^{n^2}-2^n} \leq \left| \frac{(v_j-v_i)}{(v_\ell-v_k)} \right| \leq \frac{2^{n^2}-2^n}{2^{n^2}-2^{n^2-n}}$$

Since

$$\frac{2^n-2^2}{2^n-2} \cdot \frac{2^{n^2}-2^n}{2^{n^2}-2^{n^2-n}} = \frac{2^{n^2+n}-4\cdot 2^{n^2}-2^{2n}+4\cdot 2^n}{2^{n^2+n}-3\cdot 2^{n^2}-2\cdot 2^{n^2-n}}$$

is less than 1 for $n \geq 3$,

$$\frac{2^n-2^2}{2^n-2} < \frac{2^{n^2}-2^{n^2-n}}{2^{n^2}-2^n} \quad , \quad \frac{2^{n^2}-2^n}{2^{n^2}-2^{n^2-n}} < \frac{2^n-2}{2^n-2^2}$$

and from the bounds previously found,

$$\left| \frac{(u_j-u_i)}{(u_\ell-u_k)} \right| \neq \left| \frac{(v_j-v_i)}{(v_\ell-v_k)} \right|$$

Now suppose $\max(v_j,v_i) \neq \max(v_\ell,v_k)$.

Then it can be shown that

$$\left| \frac{(v_j-v_i)}{(v_\ell-v_k)} \right| \leq \frac{2^{n^2-n}-2^n}{2^{n^2}-2^{n^2-n}} \quad \text{if } \max(v_j,v_i) < \max(v_\ell,v_k)$$

or
$$\left| \frac{(v_j-v_i)}{(v_\ell-v_k)} \right| \geq \frac{2^{n^2}-2^{n^2-n}}{2^{n^2-n}-2^n} \quad \text{if } \max(v_j,v_i) > \max(v_\ell,v_k)$$

Since

$$\frac{2^2-2}{2^n-2} \cdot \frac{2^{n^2}-2^{n^2-n}}{2^{n^2-n}-2^n} = 2 \cdot \frac{2^{n^2}-2^{n^2-n}}{2^{n^2}-2\cdot 2^{n^2-n}-2^{2n}+2\cdot 2^n}$$

is greater than 1 for $n \geq 3$,

$$\frac{2^{n^2-n}-2^n}{2^{n^2}-2^{n^2-n}} < \frac{2^2-2}{2^n-2} \; , \; \frac{2^n-2}{2^2-2} < \frac{2^{n^2}-2^{n^2-n}}{2^{n^2-n}-2^n}$$

and once again

$$\left| \frac{(u_j-u_i)}{(u_\ell-u_k)} \right| \neq \left| \frac{(v_j-v_i)}{(v_\ell-v_k)} \right|$$

so that

$$\frac{(u_j-u_i)}{(u_\ell-u_k)} \neq \frac{(v_j-v_i)}{(v_\ell-v_k)}$$

The foregoing argument establishes that x lies on exactly one hyperplane H_{ij}. As a result, there exists an ϵ-neighborhood $N_\epsilon(x)$ of x such that for all y in this neighborhood, y is in A if the signs of y_j-y_i and v_j-v_i agree, and y is in A_P-A if not. Without loss of generality, let $v_j-v_i < 0$, so that for all $y \in N_\epsilon(x)$, $y \in A$ if $y_i < y_j$ and $y \in A_P-A$ if $y_i > y_j$. Since there exist vectors y satisfying this latter condition, iPj is false; by definition of P, there exists a $z \in A$ so that $z_i > z_j$. A diagram of the situation is shown in Figure 4.2.1

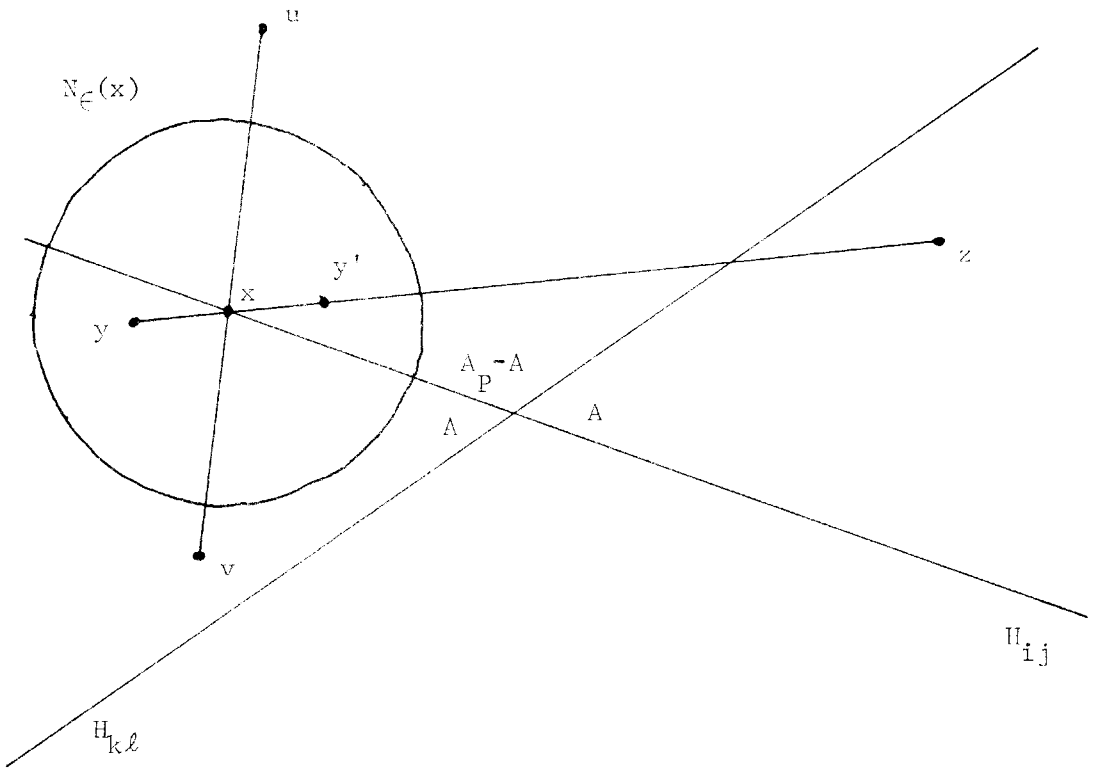


Figure 4.2.1 Situation resulting from the assumption $I_p - I \neq \emptyset$.

A contradiction will be reached by showing that there exists a $y \in A$ so that some point y' on the line segment joining y and z through x does not lie in A. Since A is convex, this is impossible, so the selection of u was impossible and $P \in T \rightarrow T \in \mathcal{T}$, i.e. $I = I_p$. It remains only to find y and y'.

Let α denote the distance between x and z, i.e.

$$\alpha = \| x-z \| = \left[\sum_{i=1}^{n} (x_i - z_i)^2 \right]^{1/2}$$

and let

$$y = (1 + \frac{\epsilon}{2\alpha})x + (\frac{\epsilon}{2\alpha})z$$

Now

$$\| x-y \| = \| -(\frac{\epsilon}{2\alpha})x + (\frac{\epsilon}{2\alpha})z \| = \frac{\epsilon}{2\alpha} \| z-x \| = \frac{\epsilon}{2}$$

so $y \in N_\epsilon(x)$. Since $x_i = x_j$ and $z_i > z_j$, $y_i < y_j$ and $y \in A$. On the other hand,

$$y' = (1 - \frac{\epsilon}{2\alpha})x + (\frac{\epsilon}{2\alpha})z$$

satisfies

$$\| x-y' \| = \left\| (\frac{\epsilon}{2\alpha})x - (\frac{\epsilon}{2\alpha})z \right\| = \frac{\epsilon}{2\alpha} \| x-z \| = \frac{\epsilon}{2}$$

so that $y' \in N_\epsilon(x)$; since $x_i = x_j$ and $z_i > z_j$, $y'_i > y'_j$ and $y' \in A_P - A$. But

$$y' = (1 - \frac{\epsilon}{2\alpha})x + (\frac{\epsilon}{2\alpha})z$$

$$= \frac{(1 - \frac{\epsilon}{2\alpha})}{(1 + \frac{\epsilon}{2\alpha})} y + \frac{(1 - \frac{\epsilon}{2\alpha})}{(1 + \frac{\epsilon}{2\alpha})} \cdot \frac{\epsilon}{2\alpha} z + (\frac{\epsilon}{2\alpha})z$$

$$= \frac{(1 - \frac{\epsilon}{2\alpha})}{(1 + \frac{\epsilon}{2\alpha})} y + \frac{\frac{\epsilon}{\alpha}}{(1 + \frac{\epsilon}{2\alpha})} z$$

and y' is a convex combination of y and z, contradicting the convexity of A.

It has been shown that if A is convex, $I = I_p$. By Theorems 3.4.11 and 3.4.10, $Z = \theta(I) = \theta(I_p) = Z_p$, and it remains to show $A_p \subseteq A$. Note first that for any real number r, the vector $r^{[n]}$, defined by $r_i^{[n]} = r$ for $i = 1,2,\ldots n$, belongs to A. Also, A is closed under positive combination; that is, for any vectors $u, v \in A$ and any positive real numbers r and s, $w = ru + sv \in A$. This result is based on the fact that

$(\frac{1}{r+s})w = (\frac{r}{r+s})u + (\frac{s}{r+s})v$ is in A by convexity, and since $r+s > 0$,

Theorem 2.1.3 guarantees $w \in A$. Now let $x \in A_p$. Clearly $\theta(x) \subseteq Z_p$, so $\theta(x) \subseteq A$. w can be written

$$w = [\min\{w_i | i=1,2,\ldots n\}]^{[n]}$$

$$+ \sum_{\substack{z \in w \\ z \neq 0^n, 1^n}} z \cdot \left[\min\{w_i | z_i = 1\} - \max\{w_i | z_i = 0\}\right]$$

Now $[\min\{w_i | i = 1,2,\ldots n\}]^{[n]} \in A$. Also $\min\{w_i | z_i=1\} > \max\{w_i | z_i=0\}$ because $z \in \theta(w)$. Since w has been written as a positive combination of elements of A, $w \in A$ and $A_p \subseteq A$. Therefore $A_p = A$. □

CHAPTER 5

CONCLUSIONS

The analysis of sorting networks appears to be a complex and difficult subject. The results of this thesis reflect the kinds of complexity that occur; most often, the problem is one of succinct characterization of the phenomenon to be studied. For example, the Boolean expressions of Section 3.2 are quite compact in contrast to the partial order theoretic characterizations of output behavior found in Section 3.3, but in fact these expressions are not nearly as economical as one might desire for a domain of eight elements, say.

Another source of difficulty is the seeming lack of algebraic structure on the objects of interest: compositions of comparators, sets of assignments, and so forth. There is at least one significant exception to this generalization, however. When the set of injective assignments that can appear at the output of a comparator network is precisely the set of injective assignments consistent with a single partial order, then many structural regularities occur. For example, it can be shown that the number of injective assignments consistent with a partial order P is a multiple of the number of symmetries of P - i.e., the number of permutations π on D such that $\pi^{-1} \cdot P . \pi = P$. Other examples of structural regularity arising from the single partial order case may be found in Section 3.4 and in Chapter 4.

Aside from the difficulties, there are two interesting phenomena that deserve mention. The first is that as a tool for network design, the Boolean expression characterization for sets of zero-one valued output assignments is much more satisfactory than the partial order theoretic characterization for injective output assignments. The manipulations required are simpler to perform, and it is easier to determine a minimal Boolean expression for a particular set of zero-one valued assignments than to determine a minimal family of partial orders by means of Theorem 3.3.3 in the injective case. Also, Theorem 3.4.14 and Example 3.4.15 indicate that sets of zero-one valued network output assignments are characterized by single partial orders more frequently than the injective output assignments are.

The other phenomenon is that in establishing properties of sorting networks, it is often useful to go back and forth from injective assignments to zero-one valued assignments. This kind of interplay occurs in Section 3.4 and especially in the proof of Theorem 4.2.3. This phenomenon is probably due to the fact that for any injective assignment there is a unique total order with which it is consistent, so that the relationship between a zero-one valued assignment and a partial order with which it is consistent is sometimes best explored by finding an injective assignment which has the zero-one valued assignment as a threshold.

Insofar as directions for future research are concerned, it is probably fair to say that no direction seems particularly promising for a resolution of the outstanding open problem about sorting networks, namely, how many comparators suffice to construct one. It would be desirable to know more about the kinds of sets of assignments that can occur as network outputs and the kinds of sets of assignments that can be assignments that are sorted by a network, but it is likely that a successful resolution of the question of the number of comparators will be based on considerations as yet unexplored.

The possibility of new techniques that will be useful for designing good networks for particular domain sizes is somewhat more hopeful. The results of Section 3.2 provide a way of exploring this area in an efficient manner. The idea is to start with the set $Z_{=}$ and apply comparators in sequence to obtain sets of zero-one valued assignments that are progressively "closer" to a sorted set of zero-one valued assignments. At each stage in the construction, the degree of "closeness" to the sorted set could be evaluated by the number of assignments in the set, the form of the partial orders that generate its filter, and so forth. It would probably be desirable to use product of sums expressions for an evaluation of the "state of the sort" and sum of products expressions for evaluating the effect of a comparator on the current set of assignments because of Theorem 3.2.3.

BIBLIOGRAPHY

Gale, David and Richard M. Karp, A Phenomenon in the Theory of Sorting, IEEE Conference Record of 1970 Eleventh Annual Symposium on Switching and Automata Theory, pp 51-59, October 1970.

Knuth, Donald E., The Art of Computer Programming, Volume 3, (Addison-Wesley, Reading, Mass.), (to be published).

Liu, C. L., Construction of Sorting Plans, Theory of Machines and Computations, pp 87-98, Edited by Z. Kohavi and A. Paz,(Academic Press, New York) 1971.

Miller, Raymond E., Switching Theory, Vol. 1, Combinational Circuits, (John Wiley and Sons, Inc., New York) 1965.

VanVoorhis, David C., A Lower Bound for Sorting Networks that use the Divide-Sort-Merge Strategy, Stanford Digital Systems Laboratory, Technical Report No. 17,(Stanford University, Stanford Calif) August 1971.

BIOGRAPHICAL NOTE

Burton J. Smith was born in Chapel Hill, North Carolina on March 21, 1941. He attended Highland High School in Albuquerque, New Mexico and the Cate School in Carpinteria, California, graduating from the latter in 1968. He spent one year at Pomona College and one year at the University of New Mexico; in June, 1960 he enlisted in the United States Navy. He served aboard the U.S.S. Triton and was honorably discharged with the rank of Communications Technician Second Class in June, 1964. He returned to the University of New Mexico and received the degree of Bachelor of Science in Electrical Engineering (Summa Cum Laude) in June 1967.

The author began studies in the Electrical Engineering Department at the Massachusetts Institute of Technology in 1967 and received the Master of Science degree in June 1968 and the Electrical Engineer degree in June 1969. He was a National Science Foundation Graduate Fellow in 1968 and a National Science Foundation Summer Trainee in 1969. In 1970 he was appointed an Instructor in the Electrical Engineering Department; he received the Carlton E. Tucker teaching award in May 1971. He is a member of Sigma Xi and Eta Kappa Nu.

In December 1966, he married the former Dorothy Nan Duncan of Nashville, Tennessee. They have a daughter, Katherine Page, born September 12, 1971.

He has accepted an appointment as Assistant Professor of Electrical Engineering at the University of Colorado, Denver, Colorado beginning September 1972.