1980

# Operational Analysis - An Aid to Interpretation of Measurement Data

Herb Schwetman

Report Number:
80-328

# OPERATIONAL ANALYSIS - AN AID TO INTERPRETATION OF MEASUREMENT DATA

Herb Schwetman
Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907

CSD-TR-328

# Operational Analysis - An Aid to Interpretation of Measurement Data

Herb Schwetman
Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907

## INTRODUCTION

In a recent article, Peter Denning and Jeff Buzen /DeBu78/ described operational analysis as a technique or approach to the analysis of computer system performance and as a basis for the development of models of computer systems. Since that time, there have been comments to the effect that models based on operational analysis were really equivalent to those based on the more traditional stochastic assumptions. Overlooked in these complaints is the contribution of operational anaylsis to the interpretation of data obtained from system measurement projects and the asymptotic bounds on performance which are readily available.

This paper presents operational analysis as an aid to interpretation of data derrived from computer systems and as a way of predicting, at a fairly gross level of detail, bounds on system throughput rates. Use of operational analysis is illustrated using data from two systems, a Burroughs B-6700 and a CDC 6500. The interested reader should refer to the original article by Denning and Buzen for a more rigorous derrivation of these results, plus an interesting collection of examples and analysis problems.

## OPERATIONAL VIEW OF COMPUTER SYSTEMS

A computer system can be viewed as a collection of resources and jobs or tasks which visit these resources, perhaps many times, until their processing requirements are satisfied. The resources represent processors, I/O devices, and other facilities of the system. Figure 1 is a schematic diagram of such a system. In Figure 1, each small box represents a system resource. The arrows represent paths which tasks follow as they circulate around the system. The arrows entering and leaving the system box represent activation of new jobs and departure of completed jobs. Typically, a job scheduler controls the allocation of long term resources such as magnetic tape drives and regions or slots in main memory

and activates jobs according to the scheduling policies of the system. Once jobs are active, they compete for use of short term resources (the processors and I/O devices of the system).

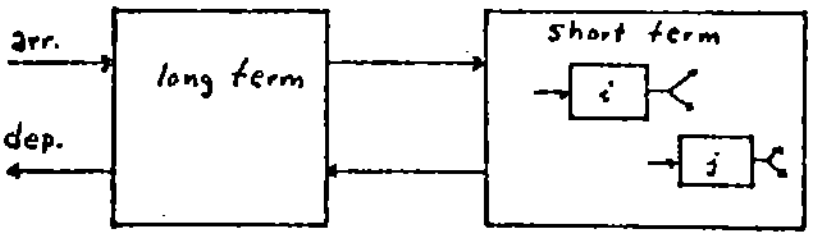


Figure 1
Schematic Diagram of a System

Operational analysis focuses on the use of the processors and I/O devices. Figure 2 shows a typical resource, numbered by the index i.

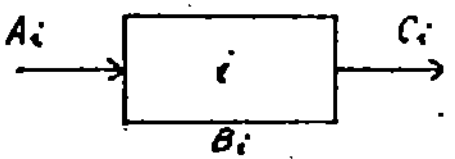


Figure 2
A System Resource

We can perform an experiment by observing this resource (within the system) for a period of time T and recording the number of job arrivals at this resource Ai, the number of departures from this resource Ci, and the total amount of time that this resource was being used by jobs Bi. Given these data, we can then calculate Si, the mean service interval, Xi, the device throughput rate and Ui, the utilization factor; these are shown in Table 1. If the data gatherer in the system can measure Wi, the total task waiting time at resource i (defined to be the sum of the queueing time and the service interval for each request), then additional performance values can be calculated, as shown in Table 2.

T = length of experiment
Ai = number of arrivals
Ci = number of departures
Bi = total busy time

Si = Bi/Ci = mean service interval
Xi = Ci/T = device throughput rate
Ui = Bi/T = utilization

Table 1
Data and Derived Values for Resource i

Wi = total waiting time

Ri = Wi/Ci = mean device response time
ni = Wi/T = mean device queue length

Table 2
Additional Performance Values

These performance values are related by the following equations:

Ui = SiXi (Throughput Law), and
ni = RiXi (Little's Law).

We can allow the index value i = 0 to represent the "outside world", the place which generates job arrivals and consumes completed tasks. Thus, A0 and C0 are the number of arrivals and departures for the entire system, and

X0 = C0/T

is the system throughput rate (the job or task processing rate). We proceed on the assumption that X0 is a useful measure of system performance.

The remainder of this paper is devoted to deriving relationships between parameters which describe the system, the jobs and this system throughput rate. This type of analysis can be extended to produce information about system response time (as will be shown in a later section).

The visit ratios,

$$Vi = Ci/C0$$

give the average number of completed visits to resource i per completed task. Then

$$Di = ViSi,$$

the product of the average number of visits to resource i and the average time per visit, is the average demand for resource i per completed job. For example, the sum of the Di's is an estimate of the time required for a single job to be processed in an empty system (i.e. with no queueing delays at the resources). Also, the reciprocal of this processing time estimates the job-processing rate when the system is operating in a monoprogrammed fashion, i.e.

$$R(1) = \text{sum } Di$$
$$X0(1) = 1/R(1),$$

where the number in parantheses denotes the level of multiprogramming, in this case 1.

While monoprogrammed operation represents one extreme in system performance (hopefully the low extreme), increasing the level of multiprogramming (the number of simultaneously active jobs) to the point of system saturation represents the other extreme in system performance. If we can characterize performance of a system operating at saturation, then we have provided a pair of bounds on system performance.

A resource is saturated when its utilization is close to 1. Using the Throughput Law, we can show that

$$Xi = Ui/Si \leq 1/Si$$

in saturation. For multi-server resources (e.g. in a twin CPU system, the CPU is usually a two-server resource), a slightly different relationship exists. If resource i has Ki servers, then Ui approaches Ki as the resource becomes saturated; in this case

$$Xi = Ui/Si \leq Ki/Si.$$

In other words, Ki/Si is the maximum request-processing rate for resource i. Using the definitions previously given, we have

$$C0 = Ci/Vi = XiT/Vi, \text{ and}$$
$$X0 = C0/T = Xi/Vi \leq Ki/ViSi, \text{ for all } i.$$

Since this relation ($X0 \leq Ki/ViSi$) holds for all resources,

the saturating resource is the one for which Ki/ViSi is the the minimum for all such values. This resource or device is called a bottleneck and will be denoted by the index b. To summarize, the bottleneck resource is the one with the smallest maximum processing rate. All of these relations are summarized in Table 3.

| | |
|---|---|
| Vi = Ci/CO | = number of visits to device i per job |
| Di = ViSi | = demand for device i per job |
| R(1) = sum Di | = processing time for a single job, single-job mode |
| XO(1) = 1/R(1) | = system throughput rate, single-job mode |
| Ki = | number of servers at device i |
| KB/VbSb = min(K1/V1S1,...), | b is index of bottleneck device |
| XO(n) $\leq$ Kb/VbSb = | upper bound on system throughput rate |

Table 3
Derivation of Performance Bounds

All of these relations are illustrated by the graph in Figure 3.

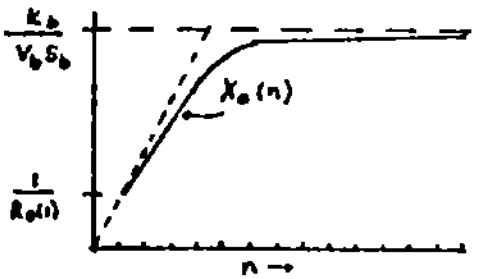


Figure 3
Asymptotic Bounds on System Throughput

B-6700 EXAMPLE

The system depicted in Figure 4 is the Burroughs B-6700 twin CPU system in use at the Computer Center of the University of Helsinki. While there are other I/O devices on the system, the ones shown in Figure 4 are the one which are critical to the processing of jobs. In this discussion, there is no description of the B-6700 hardware or the MCP operating system; the interested reader is refered to the relevant

Burroughs manuals and the book by Organick /Orga73/. In this system, the rotating mass storage devices are currently used as described in Table 4.

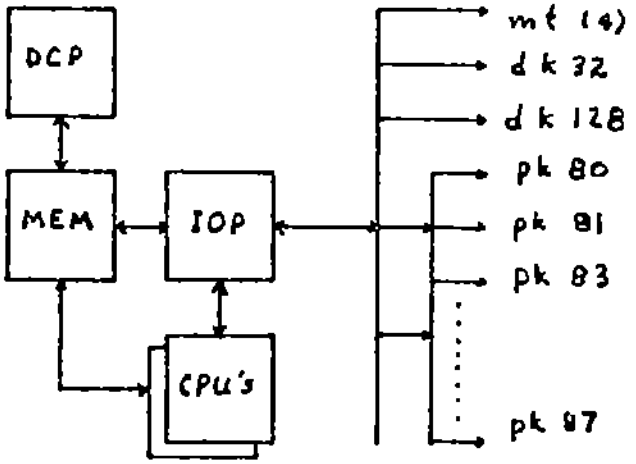


Figure 4
Schematic of B-6700 System

| | |
|---|---|
| dk32<br>dk128 | - task swapping |
| pk85 | - system libraries |
| pk86<br>pk87 | - OS modules, dynamic segment loading (paging) |
| pk80<br>pk81<br>pk83<br>pk84 | - user files |

Table 4
Uses of Disk Drives

This system supports both interactive use (via the CANDE terminal control system) as well as local and remote batch jobs. Also, the operating system has several data gathering facilities provided in the System Performance Analysis Review Kit (SPARK). One of these is SAMPLER, which can give a fairly comprehensive view of the usage of system resources. Table 5 summarizes a set of data gathered on the use of system resources during a 42 minute period of operation. The choice of devices in Table 5 was made after careful examination of the SAMPLER data. For example, the data showed that there were

never more than seven I/O requests simultaneously active; therefore there were never any delays associated with the data channels (there are 12 channels). Similarly, there were few delays tallied for any I/O control unit. Thus the choice of the individual devices in Table 5 is justified. An article by Rose /Rose78/ goes into some detail discussing issues related to developing models of real systems.

T = 2525.930 seconds

| i | | Ci | Bi |
|---|---|---|---|
| 0 | jobs | 336 | |
| 1 | CPU | 104753 | 4735 |
| 2 | mt | 332 | 106 |
| 3 | dk32 | 0 | 0 |
| 4 | dk128 | 9160 | 528 |
| 5 | pk86 | 36609 | 1786 |
| 6 | pk87 | 29652 | 1549 |
| 7 | pk85 | 8744 | 435 |
| 8 | pk80 | 4046 | 190 |
| 9 | pk81 | 4748 | 208 |
| 10 | pk83 | 3538 | 173 |
| 11 | pk84 | 7588 | 489 |

Table 5
B-6700 Data

The data in Table 5 was used to calculate performance variables according to the formulae in Tables 1 and 3; the results are shown in Table 6. The number of servers for each device is one, except for the CPU, which has two servers (K2 = 2). Further calculation shows that K2/S2V2 = .14, K3/S3V3 = .19, K6/S6V6 = .22, etc. Thus the bottleneck device is the CPU. The throughput asymptotes for this example are plotted in Figure 5.

| i | | Si | Xi | Ui | Vi | ViSi |
|---|---|---|---|---|---|---|
| 0 | jobs | | .133 | | | |
| 1 | CPU | .045 | 41.471 | 1.875 | 311.8 | 14.1 |
| 2 | mt | .319 | .131 | .042 | 1.0 | 0.3 |
| 3 | dk32 | | | | | |
| 4 | dk128 | .058 | 3.626 | .209 | 27.3 | 1.6 |
| 5 | pk86 | .049 | 14.493 | .707 | 109.0 | 5.3 |
| 6 | pk87 | .052 | 11.739 | .613 | 88.3 | 4.6 |
| 7 | pk85 | .050 | 3.462 | .172 | 26.0 | 1.3 |
| 8 | pk80 | .047 | 1.602 | .075 | 12.0 | 0.6 |
| 9 | pk81 | .044 | 1.880 | .082 | 14.1 | 0.6 |
| 10 | pk83 | .049 | 1.401 | .068 | 10.5 | 0.5 |
| 11 | pk84 | .064 | 3.004 | .194 | 22.6 | 1.5 |
| | | | | | | 30.4 |

Table 6
B-6700 Performance Variables

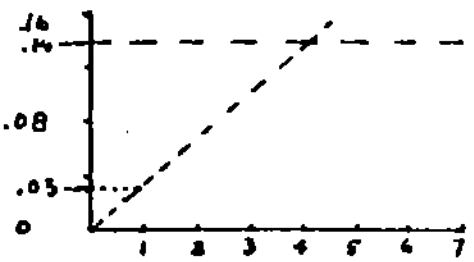


Figure 5
B-6700 Throughput Asymptotes

All of this data indicates that as the system processed this set of jobs, the CPU was the resource limiting the system throughput rate. Furthermore, as the measured throughput is .133 jobs per second (about 479 jobs per hour), the system is operating close to its theoretical maximum rate (.14). If the CPU could be speeded up, the processing rate of the pk86 device would become the next performance-limiting resource. Also, if the processing rate of a device other than the CPU is increased, there will probably be little or no effect on system performance during periods of peak demand.

As the B-6700 has an automatic segment loading feature

(which is similar to paging in virtual memory systems), the performance as a function of n, the number of active tasks, may not behave as predicted in Figure 5. More specifically, as n increases, the amount of physical memory per active task decreases, and the segment loading rate per task will increase. This will cause the Vi and Si used in the above calculation to vary; it was assumed that they would remain constant as n increased.

Further analysis of the data from the SAMPLER shows that user programs are receiving less than half of the available CPU time and that a great deal of the CPU power is being used to process segment faults. The computing center is currently formulating plans to obtain an additional block of main memory, so as to recover some of this CPU "overhead" for user programs.

## CDC 6500 EXAMPLE

The system depicted in Figure 6 is one of three CDC 6000 series systems in use at the Computing Center at Purdue University.

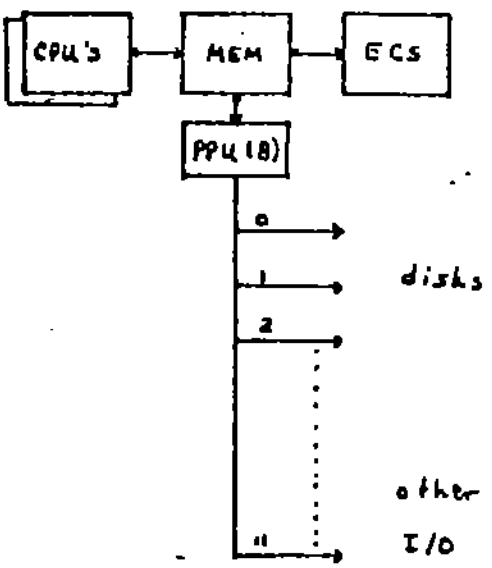


Figure 6
Schematic of CDC 6500 System

The operating system used on this system is the Purdue/MACE system, which is a highly modified version of a CDC system. One feature of the Purdue system is an event recording facility which can be invoked to gather a great deal of data about all aspects of resource usage within the system. A sample of this data , collected over a 39 minute period of operation is presented in Table 7.

T = 2353 seconds

| i | | Ci | Bi | Wi |
|---|---|---|---|---|
| 0 | jobs | 783 | | |
| 1 | CPU | 64049 | 4051 | 6898 |
| 2 | IO-0 | 1504 | 72 | 78 |
| 3 | IO-2 | 2087 | 194 | 239 |
| 4 | IO-3 | 5552 | 283 | 313 |
| 5 | IO-4 | 3552 | 980 | 1624 |
| 6 | IO-9 | 12472 | 1013 | 1760 |
| 7 | IO-T | 11599 | 569 | 832 |
| 8 | PPU(7) | 18022 | 1706 | 1706 |

Table 7
CDC 6500 Data

Again, a detailed knowledge of the system was required, in order to select a suitable set of "devices" for this analysis. In CDC 6000 systems, all I/O requests directed to a disk or tape drive are eventually processed by a peripheral processing unit (PPU). However, these PPU's can perform other processing tasks as well. The book by Thornton /Thor70/ provides a great deal of information about the hardware of these systems. In the Purdue/MACE system, most of the I/O requests are, in fact, placed by the executive in a queue, one queue for each data channel, and then they are processed, one-at-a-time, by a PPU, with one active PPU per channel queue. The goal is to achieve the maximum I/O processing rate with the minimum of wasted PPU time (i.e. to have few if any PPU's waiting to gain access to the required data channel). All of this means that the proper resources to use in representing I/O devices in a system model are these I/O queues, and not the hardware associated with the devices. The data in Table 7 was gathered from one of the 6500's at Purdue. The notation IO-i is used to denote the I/O queue associated with channel i. IO-T represents channel 11, the tape drive channel. This representation was chosen after some experimentation with several different possibilities.

The data from Table 7 was used to calculate values of selected performance variables; these are presented in Table 8.

| i | | Si | Xi | Ui | Vi | ViSi |
|---|---|---|---|---|---|---|
| 0 | jobs | | .333 | | | |
| 1 | CPU | .065 | 27.22 | 1.72 | 81.8 | 5.3 |
| 2 | IO-0 | .048 | .64 | .03 | 1.9 | .1 |
| 3 | IO-2 | .093 | .89 | .08 | 2.7 | .3 |
| 4 | IO-3 | .080 | 1.51 | .12 | 4.5 | .4 |
| 5 | IO-4 | .070 | 5.96 | .42 | 17.9 | 1.3 |
| 6 | IO-9 | .081 | 5.30 | .43 | 15.9 | 1.3 |
| 7 | IO-T | .049 | 4.93 | .24 | 14.8 | .7 |
| 8 | PPU(7) | .095 | 7.66 | .10 | 23.0 | 2.2 |
| | | | | | | 11.6 |

Table 8
CDC 6500 Performance Variables

The ratios Ki/ViSi were calculated (e.g. K1/V1S1 = .377, K5/V5S5 = .769, K8/V8S8 = 3.182, etc.) as was the monoprogrammed system throughput rate (X0(1) = 1/11.6 = 0.086). The results of these are plotted in Figure 7.

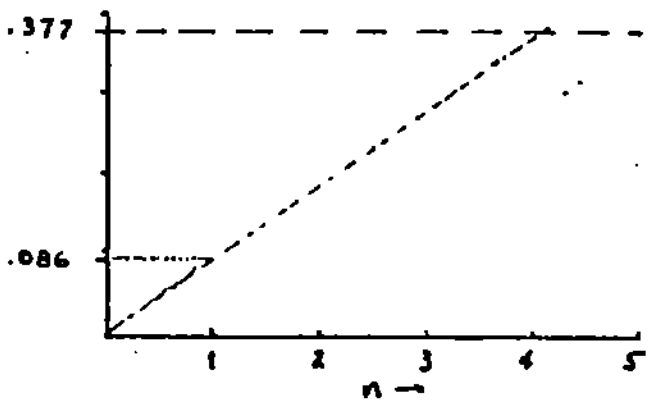


Figure 7
CDC 6500 Throughput Asymptotes

Additional analysis of these data shows that there were about 5.7 job segments simultaneously active, on the average. As the system throughput rate is measured to be .33 job-segments per seconds (1200 segments per hour), the system is operating near saturation. As with the B-6700, the CPU is the bottleneck device. However, with the 6500, the next bottleneck devices (IO-4 and IO-9) have sufficient capacity so that if the processing capacity of the CPU was increased,

total system throughput should also increase. In fact, since these data were collected, the Computing Center has installed a third mainframe (a CDC 6600) with shared access to these peripheral devices.

## RESPONSE TIME

There is an easy extension to this kind of analysis which provides similar information about the response time for a terminal system. Assume that there is such a system as shown in Figure 8. In this system, there are N active terminals; each terminal "thinks" for an average of Z units of time and then initiates a transaction to be processed by the system. The response time, R(N), is the average time required to process a transaction and then send a reply back to the originating terminal.

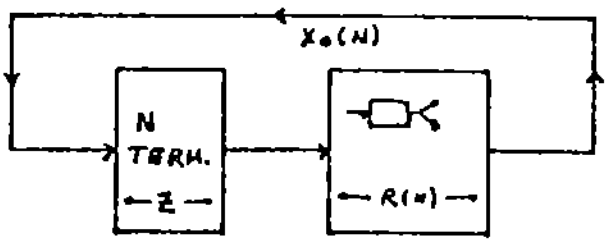


Figure 8
Terminal System

We can use Little's Law to relate N, R(N), XO(N) and Z as follows:

$$N = (Z + R(N)) * XO(N);$$

by simple manipulation, we get:

$$R(N) = N/XO(N) - Z.$$

Since by a previous result $XO(N) \leq Kb/VbSb$, we then have

$$R(N) \geq N*(VbSb/Kb) - Z.$$

Also, R(1) = sum Di, as before. Thus, we can produce a plot of response time asymptotes, as shown in Figure 9.

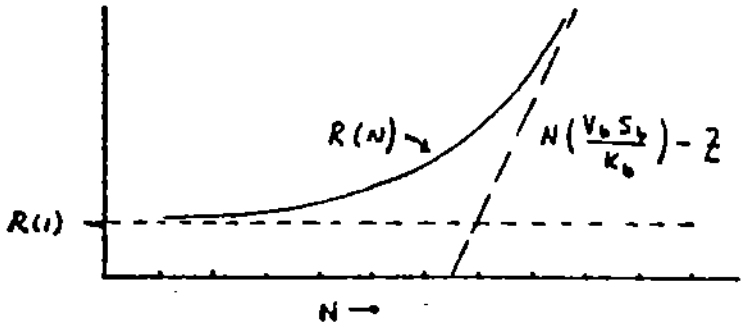


Figure 9
Response Time Asymptotes

DISCUSSION

In this paper, operational analysis, as described by Denning and Buzen, has been used as an aid to the interpretation of system data. The original article contains numerous examples of using this approach to answer questions about system performance. The article also introduces queueing network models as a natural extension of the basic ideas of operational analysis. Other articles in the September 1978 issue of Computing Surveys (ACM) discuss the formulation of system models /Rose78/ and present an example of a system model based on the MVS operating system for the IBM S/370 /Buze78/.

As the current paper concentrated on interpretation of data and on characterizing sytem performance, only a modest attempt has been made to use these techniques to predict system performance, while, for example, processing a different workload or with a different configuration. This type of prediction is more in the area of system modeling. This approach, operational analysis, to the interpretation of data allows us to conclude the following:

- The elapsed time T, the transaction counts Ci, and the device busy times Bi are the data values of importance. The total waiting time Wi can be useful. Most of the performance variables of interest can be obtained from these.

- The Throughput Law (and Little's Law) can be used to cross check the gathered data and perhaps to supply some missing items as well (e.g. to obtain the mean queue

length if only the mean response time and throughput rates are known).

- The structure of the model which represents the system is of importance; notice that we are creating a model whenever we draw a system schematic or gather data.

- Bounds on system throughput and response time are easily obtained using available data and simple calcualtions.

A note of caution should be sounded regarding the throughput rate and response time bounds. The calculations producing these are made from one set of data; as the workload varies, these data values will change, resulting in different sets of bounds. Hopefully, these will not vary too much. Also, as mentioned earlier, it is assumed that the Vi's and the Si's are independent of the level of multiprogrammed (i.e. are constants which are independent of the workload, in some sense). In virtual memory systems, this may not be true: demand for the CPU and file devices should remain the same, but the demand for the paging devices may vary significantly, as the level of multiprogramming changes, thus affecting the performance bounds.

The most appealing aspect of this approach to the interpretation of sytem data is the emphasis placed on bottleneck resources and other resources which are potential bottlenecks. The analysis technique is so simple that it can be easily performed as part of any planning project. Failure to do this could lead to the unpleasant situation in which a system upgrade does not produce the desired improvement in performance.

## ACKNOWLEDGEMENTS

This paper was written while the author was a Fullbright lecturer at the Department of Computer Sciences, The University of Helsinki, Helsinki, Finland. The assistance of Mr. Timo Saari, of the Computing Center and of Mr. Teemu Kerola (who helped formulate the material on bottleneck devices) is greatfully acknowledged.

# List of References

/Buze78/ Buzen, J. "A Queueing Network Model of MVS", Comp. Surv. (10,3), Sept. 1978, p.319.

/DeBu78/ Denning, P. and J. Buzen, "Queueing Network Models of Computer System Performance", Comp.Surv. (10,3), Sept. 1978, p. 225.

/Orga73/ Organick, E., Computer System Organization - The B5700/B6700 Series, Academic Press, 1973.

/Rose78/ Rose C., "A Measurement Procedure for Queueing Network Models of Computer Systems", Comp. Surv. (10,3), Sept. 1978, p. 263.

/Thor70/ Thornton, J., Design of a Computer: The Control Data 6600, Scott, Foresman and Co., 1970.