FORMALISM IN COMPUTER SYSTEM DESIGN
MODELS OF PARALLELISM AND CONCURRENCY
[Short Version]

Professor E. G. Coffman

Princeton University/
Computing Laboratory,
University of Newcastle upon Tyne,
Claremont Tower,
Claremont Road,
Newcastle upon Tyne, NE1 7RU.

Abstract:

The design of effective academic courses covering the topics of machine organization and operating system design constitutes an important problem in computer science curriculum development. The point of departure for the talk that was based on these notes consisted of the apparent lack of sufficient formalism in many of these courses as they are presented today. The notes proceed to show that a formal framework can be found for a wide range of subjects in the area of operating system and machine organization design. The means for demonstrating this was a survey of algebraic and stochastic models designed to represent various control and sequencing problems connected with multi-programming and multi-processing systems.

Rapporteurs:

Mr. N.S.M. Cox

Introduction

Professor Coffman indicated that he had taken his title to include the architectural design of virtual computer systems, therefore covering the design of the facility as seen by the user, and stressed that the term 'design' embraced the design of the physical device and the virtual structure created by the mapping of the operating system onto the hardware.

The main emphasis of this paper was on the mathematical modelling and formalism which should provide the framework for topics covered in courses on operating system design and the design of other computer organisations; in this respect it was stressed that the subject 'computer science' could only lay claim to the word 'science' to the extent that there existed a framework for theoretical studies and mechanisms for systematic and coherent descriptions of the many structures and algorithms arising in computer science. It was suggested that there was a frequent and unnecessary absence of theoretical framework and formal approach in many supposedly advanced courses consisting almost wholly of historical presentations of the great number of innovations present in currently active computer systems. The speaker felt that such courses could not properly be regarded as science courses, although he admitted that a certain amount of case study is required, especially in topics dealing with the inevitable ad hoc components of the internal structures of computer systems, and suggested that the time had passed when only historical presentations were inevitable. It was suggested that the existing variety in computing machines should be presented mainly to illustrate the general principles and implementation of formal, generic models where these exist, and that one of the important continuing efforts in computer education should be the devising of formal models of computations and computation structures on which systematic expositions of the techniques and algorithms connected with the computer architecture and operating systems design could be based.

The major part of this paper was concerned with illustrations, drawn from the use, implementation and control of parallelism in computer operating systems design, of the extent to which this was already possible. Professor Coffman believed that this constituted a major part of an academic - i.e. 'scientific' - course in systems design and analysis.

Machine Oriented Models

Professor Coffman described a model of modular (interleaved) memory systems, indicating that the flow of information to and from the main storage unit was a process of prime interest in the design of computer architecture. He suggested that there was frequently a disparity between achievable processor and memory speeds, making it essential to ensure that the flow of information to and from the main memory was optimized by means of organization techniques. The two dominant techniques used to increase the effective memory 'bandwidth' are memory interleaving and main storage hierarchies. It was pointed out that mathematical models of interleaved memory systems had been extensively analysed.

A model for a look-ahead instruction control unit was described and a formal program model in which the execution could involve parallelism to an arbitrary degree was discussed.

Models Oriented to Operating System Design Problems

Here, Professor Coffman was concerned with computer operations further removed from the detailed machine functions of the preceding section and his examples were concerned primarily with the study of operations expressible at the procedure - oriented statement level. Formal treatment of these processes is currently a topic of great interest, because of the framework provided for the study of the control, file structure, and scheduling problems related to large multi-programming multi-processing systems. It was stressed that the important characteristic of these computing systems was that, at any instant of time, there exist many jobs or programs whose hardware requirements are to be met by commonly available, limited resources, and whose execution is generally interdependent and in parallel. This gives rise to resource allocation problems, including the problem of avoidance of 'deadlock'. In addition, parallel execution of interacting programs gives rise to problems of mutual protection and synchronization. Solutions to these problems provide techniques and algorithms, implemented in operating systems, which themselves must be viewed in the same sense and for essentially the same purposes as the programs and system functions which they control.

In the systems considered, provision was made for specific programs to be shared by, or in execution for, more than one user at any time (i.e. a single program may constitute several processes), and for a group of

programs to be multi-plexed on a single processor. Thus, logical progress of the programs was not generally continuous with a constant set of resources and, because of the real-time and on-line aspect of these systems, programs are generally executed in such a way that the sequence and extent to which the resources are allocated to these programs are not necessarily predictable or reproducible functions of time. Thus, the notion of sequential process rather than program has come to be defined as the basic object of concern in the design problem.

Professor Coffman pointed out that the design description of the specification of resource allocation, file management and scheduling process within the operating system is facilitated and clarified by the process formalism. In particular, such a formalism provides a means for describing systems in which virtually all considerations of machine limitations and the logistics of machine operation are removed from the user/programmer, and complete program generality is provided to the user in the form of the ability to share data and procedures with other users and to construct programs from procedures (with unknown internal structure) produced by others.

Models of control problems caused by the existence of interacting processes executing in parallel were illustrated:

1. Process synchronization and mutual exclusion. The timing problem arising in operating systems design, exists in the controlling of communication of information between two asynchronous, parallel processes standing in a source-sink relationship. For example, proper control of such processes must provide a mechanism to prevent the source process from adding a new item to a full (communication) buffer and to prevent a sink process from attempting to remove an item from an empty buffer. A formal solution to this problem was outlined.

2. A storage allocation model in a paging system. The motivation for the concept of virtual memory systems, in which the programmer behaves as if the storage available to him was unbounded and at a single level, has arisen from the need to remove from the programmer the necessity to carry out resource allocation procedures arising from limited resources. Also, in an efficient realization of such systems it is generally desirable that the main storage allocations in which a process resides should not be irrevocably fixed, the system be able to

load a program even though the main storage space available to it is not contiguous, and a program be executable without the whole of that program being in main storage. The implementation of systems meeting these requirements has involved the organization of all information into fixed size 'pages' which have then been used as the 'units' of storage allocation. The one level storage seen by the program must be logically fabricated from a physical storage hierarchy and procedures must exist for the loading of programs to be executed and the handling of access to sections of that program (including its data) which are not in the main storage. A mathematical model for storage allocation with the resulting 'paging in' and 'page replacement' algorithms was described (the so-called working set model of program behaviour).

3. A resource allocation model for the study of sequencing problems. Any system, with limited resources available, permitting more than one process to execute concurrently, is subject to the problem of deadlocks in which two or more of the active tasks may not be completed because of conflicting resource requirements. Three approaches for the avoidance of deadlock are:

1. Algorithms for the detection of and recovery from deadlock (if the latter are indeed feasible);
2. The sequencing of resource usage such that deadlocks cannot occur;

and 3. The design of a system which effectively removes one of the necessary conditions for the existence of deadlocks.

A formal model of resourse utilisation by means of which the above approaches may be investigated, was given. This model permitted jobs resident in a system to be either active (in execution or ready for execution) or waiting for the acquisition of requested resources so that it can become active. Deadlocks were made possible in the model by the stipulation that resources could be released during or on the completion of a job but not while waiting for requested resources.

Job Oriented Models of Computer Operation

It was pointed out that a substantial effort has been devoted in the last several years to the study of mathematical models in which the computer is considered to be a server in a stochastic queueing system or in a job scheduling environment. The principal objectives of these studies

have been the analytic description of system performance under the existing variety of service disciplines applicable to multi-programming/multi-processing systems, and the derivation and evaluation of job sequencing algorithms designed to optimise some measure of system performance.

Almost invariably the probability models have only represented the system in a coarse fashion, modelling only the structure of a system and its operation, thus providing results which are seldom usable as a means for the measurement of the performance of that system. The value of these studies is in the means they make available to the system designer for the examination of the general behaviour of a system under the control of different algorithms and the changes in this behaviour with changes in the values of structural parameters. The analysis of such mathematical models also provides a means for the verification of Monte Carlo simulations in which significantly more system detail can be represented with less idealised probability distributions, and can often provide bounds or optima against which the behaviour of more realistic algorithms can be compared. Thus, theoretical results are valuable for use in engineering problems to provide insight and frames of reference but rarely do they yield adequate measurements for system performance.

In practice, certain properties of the mathematical models frequently place extreme difficulties in the way of a complete analysis, including:

1. Inadequacy of the exponential assumptions for inter-event times;
2. Statistical dependence between certain 'random' variables whose behaviour is to be investigated;
3. The requirement for a model of some aspect of program behaviour which is difficult, if not impossible, to formulate satisfactorily;
4. The system characteristics requiring the assumption, in a queueing model, of a finite source or bounded queue size, and
5. A model structure involving queues in series or in parallel.

Professor Coffman pointed out that the difficulties facing the study of sequencing algorithms for various job and system models have stemmed, quite simply, from the combinatorial complexity arising from the structure of the problem but emphasised that analysis had been carried out, despite these difficulties, for a considerable number of interesting problems and that idealisation, based on judicious approximation, had led to some very successful studies.

Unfortunately, the application of classical techniques has rarely been successful in establishing explicit expressions for the probability distribution function for the number of jobs in the system and the waiting times of these jobs, the performance measures usually sought in a general analysis of such probability models. Frequently, such analysis has been limited to applications of expected-value arguments and to the rather ponderous analysis of finite Markov chains.

As an example of the application of expected-value arguments, Professor Coffman described a round-robin model in which the simplest variation assumed a Poisson input with exponential service times. He pointed out that attempts to establish waiting time distributions conditional on service time required had been unsuccessful. An example in which approximation was successfully used consisted of a disc system in which two serial queues are assumed --- one for the disc arm and one for the appropriate cylinder --- the approximation arising in the assumption that the output of the first queue, which is the input to the second queue, assumes a Poisson distribution.

Two further mathematical models of computer operations were illustrated, in one of which the transform of the conditional waiting time distribution being found and in the other an optimal algorithm being established for a deterministic sequencing problem.

SUMMARY

The need for formal models of computer hardware and software systems has been emphasised in order that the basic principles of the subject can be presented in a systematic fashion without it being necessary to present the detail of the real system in which these principles are implemented. An explicit course outline has not been provided; however, the speaker suggested that it was now possible to present a major portion of a course devoted to the principles of machine organisation and operating system design in a more formal manner than is generally the case. For such a course it is clearly desirable to have the preparation in the elements of algebra and mathematical logic normally applicable to courses in automata theory. In addition, a course in probability theory and its applications is an essential pre-requisite as it should be for courses in data structures and should include, if possible, an introduction to applications in queueing theory.