Entities, functions and binary relations: steps to a conceptual schema

M. J. R. Shave

Computer Science Department, University of Bristol, School of Mathematics, University Walk, Bristol BS8 1TW and CACI Inc-International, 289 High Holborn, London WC1V 7HX

A DBMS-independent method of data analysis, which has been used with considerable practical success, is described and assessed in relation to some more formal approaches to data modelling.

(Received November 1978; revised August 1980)

1. Introduction

Considerable experience has now been gained in the use of a wide variety of systems of data base management. Ironically this very experience had led to an awareness of the benefits to be gained from a data model which is independent of any one such system, and, indeed, independent of any type of system—hierarchical, network, or relational. The objective of this model has been variously called a conceptual schema, an entity relationship model, an enterprise description, and a role model. In each case, the aim is to describe the items and relationships of the data 'as they are' in the perception of the organisation involved, and not 'as they have to be' in order to conform to the rules of a particular system of implementation.

Sections 2 to 6 of this paper describe a practical DBMS-independent approach to data analysis which can subsequently be mapped on to existing models of data systems, whether hierarchical, network, or relational in form. The analysis is carried out in two phases, known as entity analysis and functional analysis respectively, whose objectives can be briefly described as the determination of the fundamental objects of the system under investigation, and of the manner in which they are used. The principles of the method resemble those of the entity-relationship model of Chen (1976), though it is presented as a practical tool rather than a formal model. A comparison is made with this and certain other models in Section 7 of the paper.

The entity-functional method is being developed and used extensively by CACI Inc-International, a consultancy company specialising in data analysis and data base implementation. In this paper the conventions used in the diagrams are those devised by Ian Palmer and Geoffrey Baker of CACI. The opinions expressed in Sections 7 and 8 are those of the author.

2. Entity analysis

The first step is to attempt to identify all the different types of entity or basic object which are involved in the system. An *entity type* is a generic term representing a set of items (which may be empty); each item of the set is one *occurrence* of the entity type. Entity types provide a classification of the items in a system, but are not themselves classified. Thus entity types can represent widely different sets of items, such as people, objects, concepts, or even events—for example the types Student, Book, Course, Registration.

The occurrences of an entity type will in general be distinguished by the distinctive values of certain properties. Each entity type will possess one or more characteristic *attributes* whose differing values provide one means of identifying the separate occurrences of that type—for example, a student name, a book number, a course code or a registration year.

The next task is to consider whether a relationship exists between a pair of identified entity types. The relationship may once again take many different forms and its identification simply reflects some natural structure in the data. The connection may involve, for example, ownership, propinquity, similarity, structure or sequence. More than one relationship type can exist between two entity types (e.g. the entity types House and Person can be related by Ownership and/or by Occupation). A relationship may also involve only one entity type (e.g. an involuted relationship Sibling between two occurrences of the entity type Person). Each relationship has a degree, which may be (1, 1), (1, *n*) in either direction, or (*m*, *n*). As for entities, relationship types are identified at a generic level, and an occurrence of a relationship type is an ordered pair of entity occurrences which satisfy the definition of the relationship.

Having identified a set of n entity types, up to $\frac{1}{2}n(n-1)$ relationship types could in theory exist, even if involuted and multiple relationships were excluded. In practice it will be obvious that many entity types are quite unrelated; furthermore, the objective is to record only *direct* relationships. For instance, direct relationships exist between the entities Parent and Child, and between Child and School, but the relationship between Parent and School is indirect—it exists only by virtue of the child. (If the parent is also a Governor of the school this is a separate relationship, which also introduces the question of a 'role'. This is discussed later.)

3. Entity, attribute and relationship definitions

The objects and concepts of a system do not fall irrevocably into one of the three categories of entity, attribute or relationship type. A classic example is provided by data which refers to marriages. The concept of marriage could be regarded as an entity type (with attributes such as date, place, name of bride and bridegroom) or as an attribute type (a status associated with the entity type Person), or as a relationship type (connecting occurrences of the entity types Man and Woman). One of the tasks of the data analyst is to decide which of these viewpoints is the most appropriate within the system he is considering.

This intrusion of system semantics into the modelling process is perfectly proper; it means only that the model is being developed within its overall context, not that it has become application dependent. Any object may appear in different roles according to the viewpoint from which it is observed and the objective of data analysis is to avoid representing the object in the model in a way which is applicable only to one aspect or application of the system. In this respect an entity type provides the most flexible form of definition.

It will be apparent that the definitions which are formulated in the initial stages of the process of entity analysis may in practice be reconsidered as the nature and scope of the data becomes more apparent. For example, Contract may initially be defined as a relationship type between the entities Customer and Company; however, the nature of a contract may determine the department to which the work is allocated and the stock required. This would imply that Contract is more suitably regarded as an entity type in its own right, participating

CCC-0010-4620/81/0024-0042 $03.00

in relationships with Customer, Company, Department, and Stock.

Space does not permit a detailed discussion in this paper of the basis for distinguishing entity, attribute, and relationship types. However, it is important to decide on, and work to, a suitable 'level' in modelling the system, in which the fundamental concepts are not obscured by too much detail. Thus 'address' may be defined as an attribute of an entity type Person, even though it has, at a more detailed level, its own properties such as street, town, and postcode. If, however, the area in which a person lives is an important feature (for example, for a district nurse) then the area may be recorded as a distinct entity. Since the descriptive properties of attributes are defined in this model only in association with entity types, a finer level of detail inevitably implies an increase in the number of entity types. Similar remarks apply to the definition of relationship types. For practical purposes it has been found helpful to aim initially for a level of description which will produce not more than about 50 entity types, connected by at most twice that number of relationships.

4. Entity model

Once the definitions have been agreed, they can be illustrated by constructing an *entity model* of the system. To demonstrate this, consider the following simple scenario.

A local Education Authority appoints a Governing Body for each of its colleges. Each college employs a Principal and the staff are organised into departments. Each course run by a college is the responsibility of a single department but it may be taught by more than one member of staff. Students who apply to a college must have appropriate qualifications and provide a guarantee of financial support from a parent or a local Authority. On entry, each student is allocated one member of staff as his tutor, and registers for one or more courses. For each course which he joins he is allocated a numbered textbook from a stock associated with each course.

From this information the following entity types and relationship types may be defined:

Entity types	*Relationship types and their degree*		
Local Education Authority	LEA	Governing Body	1:*n*
Governing Body	LEA	Student	1:*n*
College	Governing Body	College	1:1
Principal	College	Principal	1:1
Staff	College	Department	1:*n*
Department	College	Student	1:*n*
Course	Department	Staff	1:*n*
Student	Department	Course	1:*n*
Qualification	Staff	Course	*m*:*n*
Parent	Staff	Student	1:*n*
Registration	Course	Registration	1:*n*
Book allocation	Course	Book stock	1:*n*
Book stock	Student	Registration	1:*n*
	Student	Qualification	1:1
	Parent	Student	1:*n*
	Registration	Book allocation	1:1

Attribute types have been omitted from this example for clarity, but on the basis of the brief description above it would be reasonable to define Principal, Qualification and Book allocation as attributes of College, Student and Registration respectively, rather than as entity types.

The entity model corresponding to these definitions is shown in **Fig. 1.** An oval box is used to indicate an entity type as a distinction from the rectangular box commonly used to represent a record at the storage level. A delta notation is used for a multi-valued relationship in preference to an arrowhead in

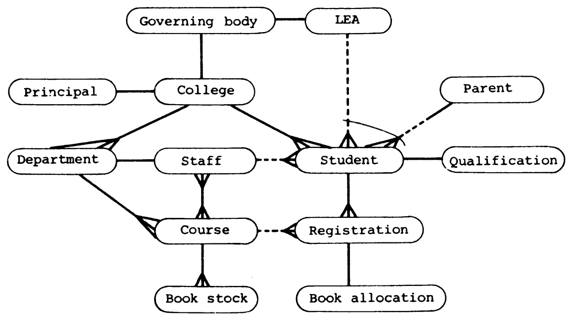

Fig. 1

order to emphasise that access to occurrences of entities may be made in either direction of the relationship. A broken line in the representation of a relationship indicates that occurrences of the adjoining entity type may exist without participating in the relationship. Thus every student must have a member of staff as his tutor, but not every member of staff will act in this capacity. By contrast, a local authority may not be sponsoring any students and also a student (e.g. a foreign student) may not be sponsored by any local authority; the arc in the diagram shows, however, that every student must have a sponsored relationship with either a local authority or a parent.

In practice, before a model such as this is constructed, a more detailed knowledge of the system would be obtained by discussion than is represented by the brief description given above. This would help to clarify the semantics of the data; whether, for example, the relationship between a college and members of its staff always occurs via their department, as shown, or whether a further direct relationship should be included in the model.

Attributes, even where they are defined, are commonly omitted from the entity model diagram since they do not add to an appreciation of its structure. The entity model *as a diagram* is simply a crude, though valuable, means of demonstrating and understanding the model of the system which has been constructed. However it also exists in a complete and precise form by means of documentation which is completed in conjunction with the diagram and which includes the definition of each entity, attribute and relationship type. As much subsidiary information as possible is recorded with each definition, such as the frequency of occurrences, period of validity, department or individual responsible for the definition, etc. . . .

5. Functional analysis

In carrying out data analysis it is possible to draw a distinction between basic objects and their structure on the one hand, and the mode of use of these objects and relationships on the other hand. The former viewpoint is reflected in the entity model, while the latter is the operational or functional aspect of the system which has not as yet been considered in detail. The distinction is not entirely clear-cut—for example, operational considerations may determine whether an entity definition or a relationship definition is the more appropriate for some concept, such as 'tutor'—but it is nevertheless useful and worth attempting.

The functional viewpoint is inevitably more application dependent than the phase of entity analysis and should therefore play a secondary role in the construction of a conceptual model. Yet it is the tasks of a system which make its existence meaningful, and thus functional analysis will often throw fresh light on the semantics of the data in the entity model. The explicit recognition of the two phases of entity analysis and functional analysis, especially if two distinct teams of analysts are involved, working separately but in parallel, can provide

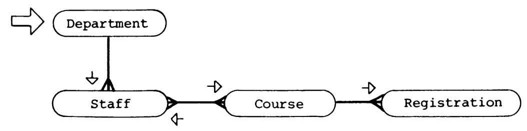

Fig. 2

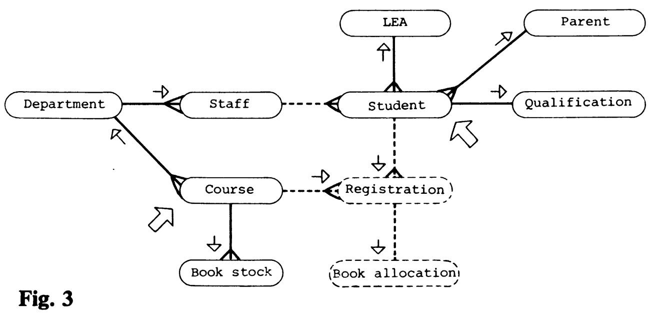

Fig. 3

valuable cross-checks on the validity, completeness and redundancy of the entity model.

The objective of the functional analysis phase is to identify which entities and relationships must be accessed, in what order, by what means (by 'owner', by key, in sequence, etc.), and for what purpose (retrieve, create, delete, update) so as to carry out the necessary tasks of the system. So far as possible, this must be done without being influenced by any existing methods of operation. The result of this analysis is documented as for entities, attributes and relationships, including any available statistics on frequency of use of the function, volumes of data involved, etc. The data structure which is accessed by each function can then be shown diagrammatically in a functional model. Each such model should appear as a subset of the entity model, subject to the addition of symbols showing the access paths required. Any discrepancies between the models must be investigated and reconciled. To illustrate functional models, consider two functions based on the previous scenario.

Function 1 *Determine the staff/student contact ratio for a given department*

The initial data is the department concerned. (This is indicated in the diagram by a fat arrow symbol.)

Thereafter it is necessary to access:

(*a*) all staff of this department

(*b*) each course taught by a member of staff

(*c*) each member of staff associated with a course (if three members are concerned, each counts only 1/3 of a unit in the staff total).

(*d*) each registration of a student for a course in the department.

This analysis can be briefly documented as follows:

Entities accessed	*Relationship used*	*Selection criteria*	*Action*
Department		Department name	Retrieve
Staff	Department/Staff	All	Retrieve
Course	Staff/Course	All	Retrieve
Staff	Staff/Course	Course code	Retrieve
Registration	Course/Registration	All	Retrieve

The corresponding functional model is shown in **Fig. 2**, where a small triangle is used to indicate the direction of access in a relationship.

Function 2 *Given the name of a new student and the course he wishes to take, check his qualifications and guarantor, create suitable registrations, and allocate to him a tutor and appropriate books.*

A brief documentation is:

Entities accessed	*Relationships used*	*Selection criteria*	*Action*
Student		Student name	Retrieve
Qualification	Student/Qualification		Retrieve
Either: Parent	Parent/Student	'Owner'	Retrieve
or: LEA	LEA/Student	'Owner'	Retrieve
Course		Course code	Retrieve
Book stock	Course/Book stock	Sequence	Modify
Registration	Course/Registration and Student/Reg.		Create
Book allocation	Registration/Book alloc.		Create
Department	Department/Course	'Owner'	Retrieve
Staff	Department/Staff	All	Retrieve
	Staff/Student	Staff load	Connect

The functional model is shown in **Fig. 3**. Those entity and relationship occurrences which are created by the function are shown with broken lines.

A practical difficulty in functional analysis is to choose a suitable level of interest at which functions are to be defined. At one extreme the entire system is a 'registration function' or a 'payroll function'. At the other extreme, too much detail may lead to an undesirable mimicry of existing processing methods. A guide can be obtained from the aim of matching functional models with the entity model. Also the level should be such that the objective of a function remains unaltered even if the method of doing it is changed. Thus one might consider a function 'book issue' which requires the identification of a book, a borrower and the date of issue, but the analysis would be likely to ignore details such as the classification of the book or the method of processing the borrower's ticket.

6. Database system models

Once the entity model has been completed and validated by functional analysis it is used as the basis for the construction of a specific Logical or Schema model. This remains largely independent of physical storage considerations but is, now, subject to the constraints of a chosen data base management system, such as DMS 1100 or Total, or to a type of system—network, hierarchical or relational. The details are beyond the scope of this paper but, for example, an (m, n) relationship between two entity types such as Staff and Course may have to be replaced by two (1, n) relationships using a newly defined entity type Lecture. 'Link' records of this type are a common means of accommodating the additional constraints imposed on the natural data structure by the syntax of programming systems.

7. Comparison with other models

The similarity between the entity-functional model described above and the entity-relationship model of Chen (1976) has already been mentioned. Both models are based on the three concepts of entity, attribute and relationship, but Chen allows a richer form of definition of relationship which includes the possibility of attributes, as for entities. Neither model allows relationships themselves to be associated, although this could be appropriate if one connection is a prerequisite for another—for example, a library book cannot be 'overdue' unless it has already been 'borrowed'.

The more formal approach of Chen's method is illustrated by the declaration statements used to describe the model and the associated data manipulation language. In this respect the entity-functional approach also insists on a clear and unambiguous documentation of its model, but does not attempt to describe its subsequent operation, which is left to the data base system modelling stage. By contrast, Chen's method does not include a phase of functional analysis, and it is true that this does avoid an element of application dependency. However this aspect takes effect only at the point of transforming the entity model into a feasible data base system model and prior to this stage the independent viewpoint provided by functional analysis forms a useful check on the accuracy of the entity model.

The so-called network model is usually taken to be synonymous with a Codasyl system (1971; 1973). As such it takes a comparatively low-level view of data which includes statements of record format and, to some extent, physical placement, so that it is not comparable to the entity-functional model. It is true that entity analysis is not based on formally defined data types and uses only simple binary relationships. But its aim is to identify the conceptual structure on which an organisation or operation is based, irrespective of the representation of those concepts, so that it is properly regarded as a high level, system independent, tool of analysis. It is noteworthy that the method of analysis mirrors the natural mode of thought: one thinks first of the basic features of a problem, and then tries to structure or organise these thoughts—a process in which binary relationships are perhaps more readily identified than those involving numerous objects.

The long-running argument between advocates of the network model and those of the relational model is often based on the false assumption that both are describing a system from the same viewpoint of data. In fact the latter model should be regarded as a predecessor to, rather than a competitor of, the network model since it aims to provide a general, application independent, description. However the manner in which domains are allocated to, or associated in, a relation is often arbitrary rather than reflecting a natural structure. For example, given the scenario previously used to illustrate the entity-functional model, suppose that two relations Course and Student are established; to which of these should the domain Book be allocated? In addition, the choice of a key value within a relation can significantly affect the process of normalisation which is an integral part of the relational technique. A number of other semantic problems have been pointed out by Schmid and Swenson (1975), and by Fagin (1977).

Although the aim of the data base approach to systems is to establish an integrated, organisation-wide body of data, it can be a weakness of the relational method that it forces (or at least encourages) an analyst to consider aggregates of data types. This not only causes conceptual difficulties when more than a few types are involved in a relation but, more importantly, there is a danger that it can imply associations between types which are not inherent in the data, or ignore others which do exist. Provided that this problem is resolved, one great merit of the relational approach is that it gives the basis for a storage independent data manipulation language, using relational algebra. In this respect it is more powerful than the entity-functional model described in this paper, which simply provides a tool for data analysis, and not for the description of operations on data. However the merits of such languages depend on the existence of implementations which permit efficient manipulation of data on a very large scale and progress in this direction has generally been slow.

Advocates of the relational model, aware of its semantic limitations, have recently proposed several modifications. One of these is that the 'value set' of a domain (a set of characters or a range of numbers) should, in suitable circumstances, be replaced by an 'entity set'. As seen by Hall *et al.* (1976), this is a set of occurrences of, say, the entity type Person. But the members of such a set are represented by internal identifiers which are allocated by the system of implementation and which remain inviolate so long as the entity occurrence continues to belong to the data. In particular, an internal identifier would be unchanged if the external 'key' of its entity (e.g. a reference number) were amended. The purpose of an entity set is to provide an invariant synonym for an external concept and to separate this synonym from any external key which may be changed at the whim of the user. This problem of changes in values used as keys is one which has caused difficulties for all data models.

A quite different view of entities in relations is taken by Boulanger and Flory (1978). They propose that each user's view of a system should be analysed and decomposed down to a (suitably defined) level of atomic attributes. This phase of analysis is then reversed, first reconciling and rationalising the attributes so as to define entities for each application and subsequently synthesising these entities into a set of relations which constitute the conceptual schema of the system. It is doubtful if an atomic level of decomposition would be helpful in practice, and in other respects the analysis phase of this method is similar in principle to that of the entity-functional model. However it is notable that the definition of a domain of a relation is once again widened to include the concept of an entity.

A second modification suggested by Hall *et al.* (1976) concerns the concept of an 'irreducible relation'. This is defined as a relation which cannot be broken by projection into several relations of smaller degree, and subsequently joined to reconstitute the original relation. They give the following example.

The relation RESULTS:

Person	*Subject*	*Position*
A	X	1
A	Y	1
B	X	3
B	Y	2
C	X	2

can be broken by projection into two smaller relations:

Courses:		*Grades:*	
Person	*Subject*	*Person*	*Position*
A	X	A	1
A	Y	B	3
B	X	B	2
B	Y	C	2
C	X		

If these two relations are now joined on their common domain, we shall obtain two tuples for which (Person, Subject, Position) equals (B, X, 2) and (B, Y, 3) respectively, in addition to those of the original relation.

A tuple in an irreducible relation represents a 'basic fact' in the sense that it cannot be broken down further without giving the possibility of deducing erroneous information. Thus irreducible relations form a desirable goal for a relational model and, because of the number of combinations of domains that will otherwise be possible, they will frequently (though not necessarily) be binary in form. However there is no canonical method for reducing a relation to this form, and no guarantee that the basic facts which are represented will correspond to relevant concepts in the external system. It is possible that irreducible relations offer a more economical form of description that the binary relationships used exclusively in the entity-functional model, but both methods rely heavily on an intuitive understanding of the data semantics and the latter seems the more straightforward approach.

In an attempt to overcome the weaknesses of network and relational models Bachman and Daya have proposed a 'role

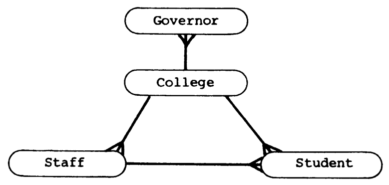

Fig. 4

model' (1977). They argue that the real world is concerned not only with basic entities but also with the roles which entity types may play. For example a 'person' can take the role of a parent, a customer, or an employee; but conversely a 'customer' can be either a person or a company. Thus neither entity types nor role types form a subset of the other. Bachman and Daya consider that failure to recognise these as distinct concepts has been the cause of considerable confusion. They propose that each entity type definition should specify one or more roles which it can support. The same role could be specified by more than one entity type definition, and the common properties of any one role would be the subject of a role type definition. An entity occurrence would thus be the union of one or more role occurrences.

The entity-functional method does not make this distinction, and the entity model which is constructed may include types which represent concepts (a role is effectively a concept) as readily as types which represent objects, people or events. The criterion for identification is simply relevance to the problem area which is being analysed. A weakness of this approach is that attributes which are common to two or more entity types are not apparent in the model (though they will be documented). For example, in **Fig. 4**, one member of staff may also be a governor of the college. (In this case the common attributes would be precisely an 'entity occurrence' of the role model.) However the model represented by Fig. 4 is a conceptual tool and the fact that two entity occurrences are unified in one person is, in this view, an 'accidental' relationship, not an inherent property of the entity types Staff and Governor. It is therefore not represented in the model. The level of implementation is a different matter. Then the common attributes must be recorded, but this can be achieved by placing their data in a separate physical record, linked to the data which is specific to 'staff' or 'governor'.

The question is thus whether the eventual implementation of the system being analysed will be best served by identifying and distinguishing in the conceptual model between roles and those objects which can adopt the defined roles, or by concentrating on the recognition of distinctive roles as concepts and postponing consideration of common properties until an implementation phase. In principle the former is the more structured and the more satisfactory approach but, as Bachman admits, there is as yet no DBMS capable of implementing a role model once it is constructed.

Another consideration is whether roles and entities can always be clearly separated. The problem is most likely to occur in the case of objects, where the dramatic analogy is less familiar. For example, a college seems to be a well defined object and hence a suitable subject for an entity type definition, but it is seen in quite different ways by its students, its staff, and its governors. Should one therefore define a number of role types (e.g. 'place of work', 'area of responsibility') to be associated with the entity College? Does 'place of work' include both the roles 'place of study' and 'place of employment'? If occurrences of these role type are infrequent in the system under review, there is a danger that there will be a proliferation of definitions and the modelling process will become unnecessarily complicated.

8. Summary

This paper has two objectives. Firstly, to describe a practical method for the analysis of data in a complex system and for the design of a coherent model of the concepts and relationships involved; secondly to place this pragmatic technique in the context of a number of formally based theoretical methods. These objectives could be said to correspond to the two most important properties of a conceptual model: simplicity of use and a sound theoretical basis (Date, 1977).

The entity-functional model uses just three data constructs—entity, attribute and relationship—and the last of these is employed only in a binary form. These constructs are not formally defined but, as 'concepts' themselves, it is difficult to do so in any way which is meaningful and not recursive. Examples have been given to show that, in formulating definitions, the most appropriate view of an item of data is not absolute and must often depend on knowledge of the data semantics. Unlike the relational model, these data constructs are not used as the basis for a data manipulation language. However the entity model is validated by a separate but parallel phase of analysis which considers the functions of the system and their impact on the data structure.

The model does not attempt to distinguish between roles and objects as has been advocated recently. It would be possible to develop the method on these lines if it was found that the resulting structure, in gaining precision, still retained clarity and simplicity. The closest theoretical parallel with the method is the entity-relationship model of Chen (1976). This allows a more detailed form of relationship and, although it has no explicit phase of functional analysis, operations on the model can be expressed in a SEQUEL-like language (Chamberlin and Boyce, 1974). However its basis is essentially the same three data constructs as used in the entity-functional method.

This latter method docs not pretend to be a complete system for data description and manipulation, but in practical applications of data base systems it has been found to provide a tool which is simple, coherent and flexible for the crucial initial stages of data analysis and conceptual design.

Acknowledgement
The entity-functional method has been developed and used by the Database and TP group of CACI Inc-International and I am most grateful to them for their permission to publish this paper. The opinions expressed in the paper, are however, entirely the responsibility of the author.

References

Bachman, C. W. and Daya, M. (1977). The Role Concept in Data Models, Infotech State of the Art Tutorial on Advances in Database Technology, London.

Boulanger, D. and Flory, A. (1978). La Notion de Type d'Entité dans le Modèle Relationnel, *Proc. des Journées d'Etude sur le thème des Modèles Relationnels*, Institut de Programmation, Université Pierre et Marie Curie, Paris, pp. 1-18.

Chamberlin, D. D. and Boyce, R. F. (1974). SEQUEL: A Structured English Query Language, IBM Research Report RJ 1394, San Jose, California.

Chen, P. P-S. (1976). The Entity-Relational Model—Towards a Unified View of Data, *ACM Trans. on Database Systems*, Vol. 1 No. 1, pp. 9-36.

Codasyl DBTG. (1971). *Report*, ACM, New York, and British Computer Society.

Codasyl DDL Committee. (1973). *DDL Journal of Development*, ACM, New York, and British Computer Society.

Date, C. J. (1977). *An Introduction to Data base Systems*, 2nd Ed, Addison-Wesley, p. 445.
Fagin, R. (1977). Multivalued Dependencies and a New Normal Form for Relational Databases, *ACM Trans. on Database Systems*, Vol. 2 No. 3, pp. 262-278.
Hall, P., Owlett, J. and Todd, S. (1976). Relations and Entities, *Proc. IFIP TC-2 Working Conference on Modelling in Data Base Management Systems*, ed. G. M. Nijssen, North-Holland, pp. 201-220.
Schmid, H. A. and Swenson, J. R. (1975). On the Semantics of the Relational Data Model, *Proc. ACM-SIGMOD International Conference on the Management of Data*, ed. W. F. King, pp. 211-223.

Book reviews

PDP-11 Assembler Language Programming and Machine Organization, by M. Singer, 1980; 178 pages. (*John Wiley*, £6·80)

This book is aimed at the reader who has no prior knowledge or experience of computers, but who wishes to 'enjoy the fullest range of communication with the machine by understanding its machine code'. While many would argue with the assumptions inherent in this aim, or indeed with the desirability of fulfilling it, the author covers his subject with an admirable clarity that does much to justify his viewpoint. By concentrating on the PDP-11 range, a thoroughly practical approach is adopted, and it is assumed that the reader has access to a PDP-11 installation on which to develop programs.

Chapter 1 introduces the PDP-11 computer system. It includes a discussion of the operating systems insofar as this is necessary to get a simple program to run. Unfortunately, despite the wise decision to narrow the book's focus to a single machine architecture, the diversity of Digital operating systems leaves the reader in considerable doubt as to how to accomplish simple tasks, and he is in danger of being overwhelmed at the outset. The author makes a valiant attempt, but inevitably fails to bring solid order to this quagmire.

Chapters 2 and 3 introduce a broad range of assembler language instructions as they are required in the process of developing working programs. The use of monitor calls for I/O is encouraged to enable real programs to be written at this stage, but a full treatment is deferred until Chapter 4. A structured approach to programming is introduced unobtrusively as a consequence of program function. This and subsequent chapters are written with a refreshing clarity, and painlessly achieve the objective of covering the entire range of instructions. The author has a useful, if slightly irritating habit of interposing a penetrating question, just as the reader's mind is glossing over a not quite obvious point. This is a real aid to fuller understanding.

Chapter 4 covers peripheral control starting with a brief, but adequate treatment of direct I/O, interrupts and UNIBUS operation. The use of monitor calls for handling storage devices is discussed, followed by an overview of their operation and control requirements. A section on memory management winds up the discussion of operating system function.

Each section of the book has a good range of exercises, though no sample solutions are provided. A description of floating point arithmetic is sensibly banished to an appendix, as is the use of the debugging tool ODT. In all, the book is a useful and readable introduction for both the newcomer to assembly language programming, and the specialist meeting the PDP-11 family for the first time.

I. M. C. Shand (Kingston)

Introduction to the Computer—An Integrated Approach, by J. Frater and W. Holdrup, 1980; 449 pages. (*Prentice-Hall*, £11·65)

The book is intended primarily to be a text for use in a one-term introduction to data processing courses and possibly in courses on computers and society. It claims that its 'integrated approach' is unique. It is certainly an approach which I believe would work well in a teaching situation in which the aim was to give students a well rounded appreciation of computers and data processing. The essence of the approach lies in the format of each chapter, in which certain new technical concepts are described followed by a substantial section on applications and implications. Thus the student is able, as he builds up his knowledge of the computer, to see in parallel something of the applications and social implications of the concepts he has mastered. Each chapter opens with a statement of the main concepts, applications and implications which are covered in it together with a list of the new terminology which it introduces. Chapters conclude with a summary and a set of about 10 relevant exercises. The book is thus well structured for use as a course text or for self-study; the preface contains helpful suggestions for both teachers and students on how it might best be used.

While it does not attempt to go into the various technical concepts in depth (it is not aimed primarily at those wishing to pursue a career in data processing), the range of topics covered is sufficiently comprehensive to enable the reader to acquire (as stated in the preface) 'the much needed computer literacy which is almost a prerequisite for entering any field'. Chapters 1-4 are concerned with basic concepts, 5-8 with hardware, 9-11 with programming and software and 12-14 with advanced systems and future trends. No specific programming language is presented in the main text but short appendices on BASIC, COBOL, FORTRAN and Pascal provide sufficient information on these languages to enable the student to write simple complete programs in any of them. The last three chapters cover such important topics of the present time as teleprocessing, networks, data base systems and the microprocessor revolution.

The book is well and accurately produced, although some of the photographs are not helpful, and the author's style is readable if a little repetitive in parts. The use of cartoons here and there provides welcome light relief and some well chosen quotations give additional insight throughout the book.

One small but important criticism: I felt that the fundamental concept of the stored program was not adequately explained early enough; I would have expected this to have been included in the first chapter. In one or two places, e.g. the chapter on 'The computer and its heritage' the writing is very much from an American standpoint, but I do not believe this necessarily invalidates the book from being usable in a British situation. Despite the length of the book some topics mentioned, e.g. computer art, are not developed sufficiently to give any real insight into them. Another general criticism is that the book does not provide directly for any practical access to a computer and the reader would certainly need to have such experience for much of the material to come alive.

In summary this is a readable and usable text, providing an all round appreciation of computers and data processing. It should certainly succeed in debunking any misconceptions the reader may have had and generate in him a moderate and critical attention to computers and their applications, convincing him that the ubiquitous computer is simply a powerful tool which man may use for good or ill as he chooses. It is a large book, perhaps a little longer and more expensive than is ideal for its purpose, but has the advantage of being more up to date than many of the alternative texts currently available.

John Lindley (Middlesbrough)