TECHNICAL REPORT 9007

Published 1987-07-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION

Information processing systems — Concepts and terminology for the conceptual schema and the information base

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work.

The main task of ISO technical committees is to prepare International Standards. In exceptional circumstances a technical committee may propose the publication of a technical report of one of the following types :

— type 1, when the necessary support within the technical committee cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development requiring wider exposure;

— type 3, when a technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical reports are accepted for publication directly by ISO Council. Technical reports types 1 and 2 are subject to review within three years of publication, to decide if they can be transformed into International Standards. Technical reports type 3 do not necessarily have to be reviewed until the data they provide is considered no longer valid or useful.

ISO/TR 9007 was prepared by Technical Committee ISO/TC 97, *Information processing systems*.

The reasons which led to the decision to publish this document in the form of a technical report type 3 are explained in the Preface.

UDC 681.3.02 **Ref. No. ISO/TR 9007 : 1987 (E)**

Descriptors: data processing, information interchange, factual data bases, basic concepts.

Printed in Switzerland

Price based on 120 pages

PREFACE.

0.1. PURPOSE OF THE REPORT.

It is expected that future data base management systems will include a component for handling conceptual schemata. This Report explains the roles and concepts for a conceptual schema with the intention of providing a framework for discussion and for the design of conceptual schema languages. The rules described in the conceptual schema control to a large extent what may or may not happen in an information system and a data base. Therefore this Report is not limiting its attention to the conceptual schema alone, but also considers basic concepts for the mechanisms involved in manipulating a conceptual schema and a data base.

This Report is aimed at designers of information systems and data bases as well as suppliers of conceptual schema facilities. The provided framework will prepare the way for eventual standardization in the area of data base management. It does not, however, describe any particular method for using such facilities. In the meantime, the general principles in this Report can be used to evaluate emerging DBMS facilities.

The approaches and associated languages described in appendices to the Report are intended to be explanatory only and are not ipso facto candidates for a standard conceptual schema language.

0.2. STRUCTURE OF THE REPORT.

The main body of the Report (chapters one through four) contains the fundamental concepts and terminology for the conceptual schema, the information base, and the mechanisms involved in manipulating them.

Chapter one gives an introduction to the subject, mentions the origins of some ideas developed in the Report, and discusses some major topics. In particular, it explains what a conceptual schema is used for, its roles, and requirements for a conceptual schema facility.

Chapter two explains fundamental concepts, provides definitions of the concepts and terms, and develops some of the consequences of those concepts and definitions. Both static and dynamic aspects of the information system are considered and explained. Some readers may wish to skip this chapter on the first reading.

Chapter three discusses some aspects of implementation. In particular, principles are formulated for the contents and scope of a conceptual schema, and an information system architecture based on three levels is outlined.

Chapter four reviews some approaches to information modelling and manipulation for data bases. The approaches selected for illustration are outlined in more detail in appendices to the Report.

Several appendices have been added to the Report as follows:

Appendix A gives a glossary of the terms and definitions.

Appendix B provides an example situation to be described in information modelling approaches.

Appendix C gives a syntax notation to be used for defining grammars of example conceptual schema languages.

Appendix D outlines Entity-Attribute-Relationship approaches.

Appendix E demonstrates Binary and Elementary N-ary Relationship approaches.

Appendix F discusses Interpreted Predicate Logic approaches.

Appendix G elaborates on expressing dynamic rules and constraints in conceptual schemata.

Appendix H presents some thought on interacting with information systems and examples of permissible action descriptions.

0.3. STATUS OF THE REPORT.

This Report is an ISO Technical Report of type 3. It is the Working Group's first response to item 1 of its Program of Work. As such it is a statement of the Working Group's current view on concepts for conceptual schemata and information bases. Considering the rapid development in data base technology and applications possible, also taking into account the requirements for distributed data base systems and related data communication facilities, periodic revisions of the Report are to be expected.

0.4. REFERENCES.

[1] MURRAY, J.A.H. et al.
'The Oxford English Dictionary' with supplements, Clarendon Press, 1933 - 1977.

TABLE OF CONTENTS.

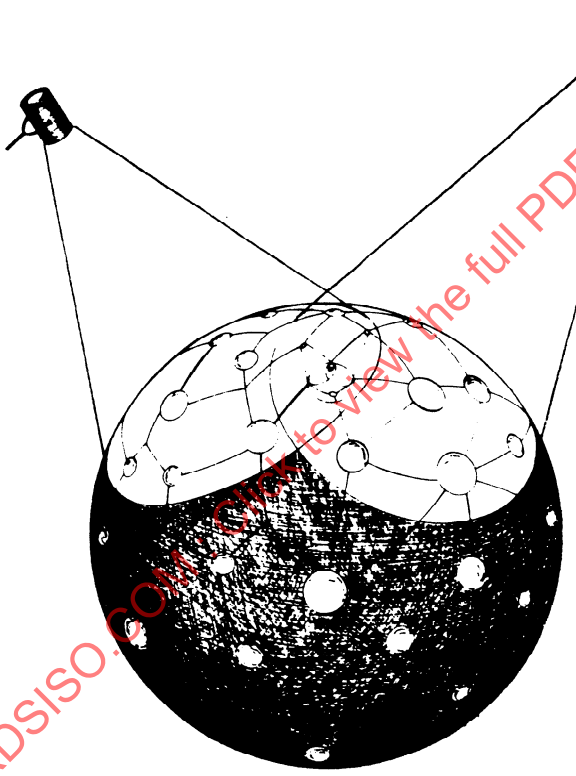
APPENDICES.

THE HELSINKI PRINCIPLE.

These utterances are to be interpreted (recursively) as international English utterances [1]:

> Any meaningful exchange of utterances depends upon the prior existence of an agreed set of semantic and syntactic rules. The recipients of the utterances must use only these rules to interpret the received utterances, if it is to mean the same as that which was meant by the utterer.

(ISO TC97/SC5/WG3 - Helsinki 1978)



"THE METAPHOR OF THE SEARCHLIGHTS" on universes of discourse.

This page intentionally left blank

CHAPTER 1. INTRODUCTION TO THE CONCEPTUAL SCHEMA AND THE INFORMATION BASE.

1.1. THE ANSI/SPARC FRAMEWORK.

The reports of the ANSI/X3/SPARC DBSG [1, 2] identified the need for a conceptual schema in the context of a three-schema framework for data base management systems.

Subsequent papers [3, 4, 5, 6, 7] have emphasized the importance of a conceptual schema to users and designers of data base systems. In this context, a conceptual schema comprises a unique central description of the various information contents that may be in a data base. This includes the description of what actions, such as changes and retrievals, are permissible on the information content. The data base itself may be implemented in any one of a number of possible ways. Users and application programs may view the data in a variety of ways, each described by an external schema. Each external schema is therefore derived from the common conceptual schema. The physical storage structure that may be in use at any given time is described by an internal schema that is also derived from the conceptual schema.

The conceptual view, as meant by ANSI/SPARC, concentrates on the meaning of the information. It is the conceptual schema that describes this view. The external views concentrate on the forms - the data - that represent the information to the outside. These are described in the external schemata. The internal view concentrates on the internal physical representation of the data inside the computer system and is described in the internal schema.

Such a three-schema framework is widely, but not yet universally, accepted. It is assumed in this report. Furthermore, it may be noted that the conceptual schema concept is valuable in other environments than a three-schema framework.

It is widely acknowledged that the conceptual schema also plays a key role in systems analysis and data base design. One may therefore ask whether it should be biased to one or the other. Should the conceptual schema be primarily an enterprise model, resulting from the systems analysis, or should it serve as a focal point between user views and the physical data base design? We believe that it should play both roles in the next generation of DBMS.

We believe the data base user will benefit from the clear separation of the information meaning from the external data representation and the internal physical data storage layout. A clear methodology for producing a conceptual schema would help the implementor of an information system to improve his systems analysis, even if a manual step of translating it into data base design in terms of an existing DBMS were then required.

The ANSI reports introduced the conceptual schema in broad terms. Besides, the term "conceptual schema" is sometimes used for data base aspects which are not at all conceptual. Therefore, elaboration of the conceptual schema's objectives, roles, form, and content is needed. What a conceptual schema must include, which appropriate modelling concepts are to be used in it, and the exact role it plays in data bases, are the major subjects of this Report.

1.2. THE UNIVERSE OF DISCOURSE.

In the past, data processing systems were often designed so as to provide all users with the same set of capabilities or functions. However, this uniform functional view is not adequate to construct today's data base systems. A single data base may support quite different functional requirements concurrently, or at different times, during its existence.

The prime characteristic of the data base environment is that common data is shared between many users of a single system. By sharing common data, these users establish a dialogue with each other through the system. Clearly, if this communication is to be useful and reliable there must be some common understanding of the information represented by the data. Since it may happen that two users never meet, this common understanding must refer to something external to both of them. This common understanding must be recorded and in order to establish a dialogue a common predefined established grammar is needed.

We will call those things and happenings to which the common understanding of the represented information refers the universe of discourse. Universes of discourse may be concrete like an inventory, or abstract like the organizational structure of an enterprise. They even may be hypothetical like Wonderland which was visited by Alice.

In this Report we will take an (informal) naive realism approach to universes of discourse.

The typical universe of discourse is perceived as containing real and abstract objects, which we will call entities. It can be perceived as also containing classes of entities, e.g. persons, departments, and dates. This classification is based on similarity and takes into account characteristics common to several entities. The selection of characteristics for grouping the entities into classes is arbitrary; the choices will be made pragmatically, based on the purpose of the universe of discourse.

Some general properties to which entities adhere, that classify entities, that associate entities, etc., in the universe of discourse are also perceived (e.g. persons are not departments, a person may be assigned to no more than one department). These may be informally described as "classifications", "rules", "laws" or "constraints" about the state of affairs and behaviour of entities in the universe of discourse.

In general, what is considered to be part of the universe of discourse will be time-dependent, that is, the selected things and happenings may change with time. This will be equally true for the classifications, rules, laws, etc; however, it is likely that the rate of change of these will be relatively slow compared with that of the former.

1.3. DESCRIBING THE UNIVERSE OF DISCOURSE.

There are in fact two systems of interest: the universe of discourse and a data processing system which contains a linguistic representation of that universe of discourse. Following common usage we say that information about the universe of discourse "describes" or "models" that universe of discourse. We want, however, to emphasize that the description process may be in fact a very complex task calling for creative analysis and iterative refinement.

Without prejudging its physical representation we consider that the information contained in the data processing system describes the universe of discourse. A concrete physical representation of this information will be called a data base. We will use the term data base system for a data processing system dealing with a data base. It is possible for the data base system itself to be one of the subjects being described, in which case the data base system would be included in the universe of discourse. However, to simplify the discussion, we will generally assume that the data base system is disjoint from the universe of discourse, although this is not necessarily the case.

It is the classifications, rules, etc., that are of primary interest to a systems designer designing a data base system. In analysing the universe of discourse, it is these things he will want to identify, discuss with users and describe. In recording them he will actually create a "skeleton" description of the universe of discourse, the conceptual schema. In this way the conceptual schema describes which entities can possibly exist in the universe of discourse, that is, which entities exist, have existed, or might ever exist. In the same sense it describes what facts and happenings are possible for those entities or, if relevant, are required for them. We assume it will be held in a formal representation within the data base system.

We also want to record all other relevant information which describes the entities that are considered to be of interest and their actual state of affairs at a specified instant or period of time (usually "now"). We call this further information the information base.

Although each description necessarily will have a representation form to make the description communicable, it is the interpretation of this representation (the meaning of the description) which interests us in the first place. The representation form, although not irrelevant, is considered to be of secondary importance. We will use the term "information" when we want to emphasize our interest in this interpretation. We will use the term "data" when we want to concentrate on the representation forms of the information.

At this point it may be useful to consider the information describing a universe of discourse within the context of an ANSI/SPARC framework: We consider both the conceptual schema and the information base to be at the conceptual level, providing a conceptual view of the information about the universe of discourse.

The data base or parts thereof as seen by a user of the system (the strings of data) we consider to be at the external level giving an external view on the information about the universe of discourse. The internal storage forms within a computer we consider to be at the internal level being the internal view of the information about the universe of discourse. For the external and internal views the representation forms are of primary interest. The interpretation of those forms is, of course, the interpretation meant in the conceptual view.

Summarizing we have now identified:

- o Universe of discourse:
  The collection of all objects (entities) that ever have been, are, or ever will be in a selected portion of a real world or postulated world of interest that is being described.

- o Conceptual schema:
  The description of the possible states of affairs of the universe of discourse including the classifications, rules, laws, etc., of the universe of discourse.

- o Information base:
  The description of the specific objects (entities) that in a specific instant or period in time are perceived to exist in the universe of discourse and their actual states of affairs that are of interest.

Precise definitions for the above concepts will be given in chapter 2.

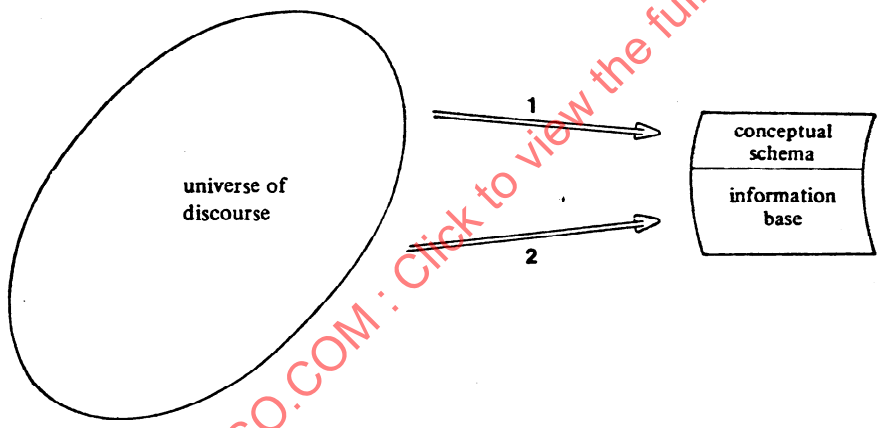


Figure 1.1. Describing the universe of discourse.

The description process is illustrated in figure 1.1; the two numbered processes are:

1: Classification, abstraction, generalization, establishing rules, etc, about the universe of discourse and recording them. This is a human process, describing a (shared) mental model of the universe of discourse.

2: Recording facts and happenings about the universe of discourse including what entities actually are of interest.

The conceptual schema describes the general rules, etc, of the universe of discourse, which, to a great extent, may govern its behaviour. These rules described in the conceptual schema therefore also control what may or may not occur in the information base. It is for this reason that we do not limit our attention to the conceptual schema and information base; we will also consider the mechanisms involved in manipulating the contents of the information base and the conceptual schema.

1.4. STATIC AND DYNAMIC ASPECTS OF A CONCEPTUAL SCHEMA AND INFORMATION BASE.

Much of the past work on concepts for the conceptual schema has been concentrated on the static aspects, that is, on defining the concepts to be used to describe valid states of a conceptual schema and information base.

However, the set of concepts for the conceptual schema should also cover the dynamic aspects. Firstly, the conceptual schema may change to correctly reflect changes in the selected portion of a real or postulated world. Secondly, dynamic aspects are involved in describing those manipulations which are needed to make known part or all of the conceptual schema and information base.

In some cases, the time scales of changes within the universe of discourse and the corresponding changes in the conceptual schema and information base need not be tied closely together: changes in the universe of discourse may be recorded in the conceptual schema and information base retrospectively and even in a different sequence. In other cases, the time scales are so closely related to each other that the conceptual schema and information base necessarily become part of the universe of discourse; especially in these cases the description of this interaction must also be part of the dynamic aspects.

No clear boundary has been defined between static and dynamic aspects, and the boundary may well be found to vary between different approaches or even to be non-existent in some cases. Some of the ideas introduced on this subject in the present Report have not yet been the subject of wide debate, but may at least serve to indicate areas deserving further study. In particular it is not clear whether different sets of concepts should be used to describe static aspects and dynamic aspects, or whether, at least for some approaches, the same set of concepts may fulfil both purposes.

1.5. INTERACTION BETWEEN THE REAL WORLD AND AN INFORMATION SYSTEM.

A conceptual schema and information base is totally static unless something operates on it to cause change. That something we will call an information processor. We will define an information system as consisting of a conceptual schema, an information base, and an information processor.

An information processor operates to produce change in the information base or conceptual schema only on receipt of a message. A message contains information and/or expresses commands. Messages originate from a part of the real world referred to as the environment, which may be disjoint from, or overlap with, the universe of discourse. On receipt of an appropriate message containing a command an information processor may also operate to make known, by means of a message, information present in the conceptual schema and information base. For further details see chapter 2.

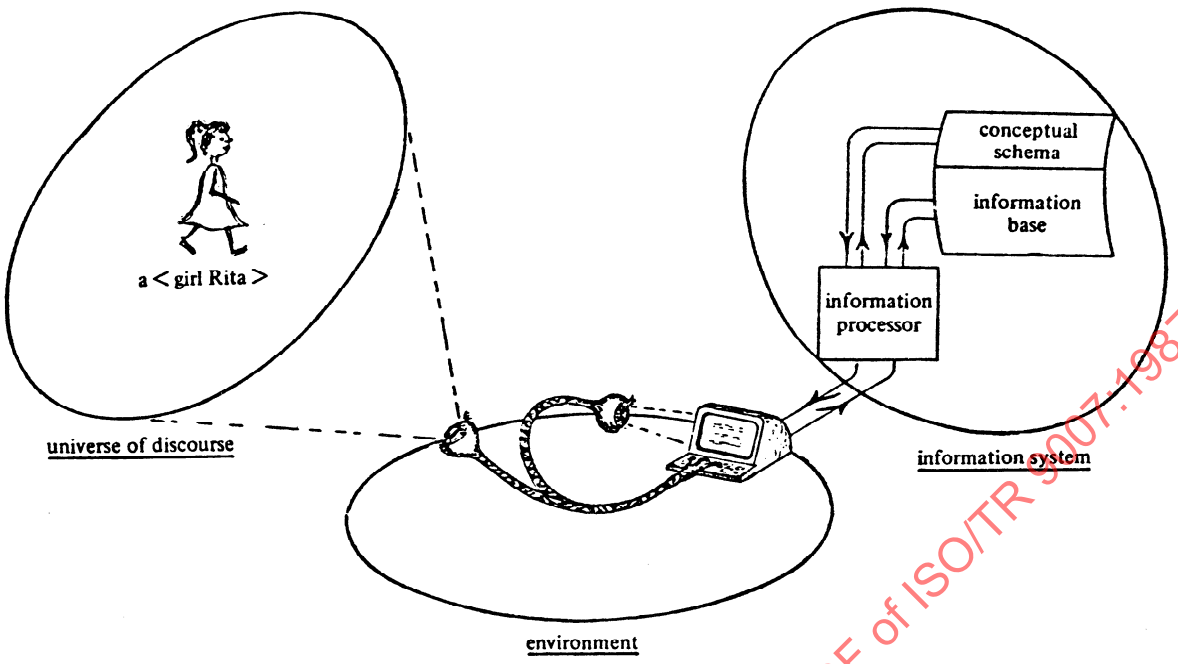


Figure 1.2. Information system and environment.

The information system is distinguished from the environment in the following way:

- o The information system is a formal system, the environment as a whole is not so.
- o The behaviour of the information system is completely defined by behaviour rules and constraints which are established, directly or indirectly, by the environment. The information system on its own initiative never establishes rules for the environment.
- o An information system, being fully predictable, is unable to deviate from the rules or constraints. The environment can deviate from its rules.

Although we may consider the information system together with the environment to be parts of an encompassing system, this latter system may not be formal or fully predictable. Therefore we use the term information system as above, excluding the environment - the users of an information system.

1.6. THE ROLE OF USERS AND INFORMATION PROCESSORS.

The users of an information system can be machines or other systems as well as human beings. A user is anybody or anything that issues commands and messages to the information system and receives messages from the information system. As such they are part of the environment. Some users also have the authority to establish behaviour rules or constraints for the information system.

An information processor transfers messages between the environment and the information base or conceptual schema, as explained above. In doing this it has no initiative of its own; it can only behave exactly as specified by the rules, the whole rules, and nothing but the rules.

Normally an information processor will be a computer system or some parts thereof, but human beings can also play the role of an information processor, provided they do not deviate from prescribed rules or act on their own initiative. Computer systems, on the other hand, can act as users of an information system. An example is a network of information systems communicating with each other. If each has a set of rules which is independent of the others, then each plays the role of user of the other information systems. We therefore conclude, that the role determines whether something must be regarded as a user or as an information processor.

The above formulation of users and information processors in terms of roles implies that the environment and the information system need not necessarily be disjoint. Similarly, if the information base contains information about the users of the information system, the environment and the universe of discourse will not be disjoint. However, even if they are not disjoint they will always be distinguishable from each other.

1.7. GUIDELINES FOR THE DESCRIPTION OF A UNIVERSE OF DISCOURSE.

Sometimes in the literature on various modelling methods for information systems and conceptual schemata, no clear distinction is made between the things and the description of the things, nor between the information meaning and the data representation.

This stems partly from the origins of some approaches which are in effect the data modelling techniques of the early seventies. Partly, however, the reason for this is a debate on fundamentals, which is still going on. The difference is found in whether the conceptual schema must be defined in terms of entities in the universe of discourse itself and states of affairs about them, or in terms of descriptive constructs found in the information base describing the entities of interest in the universe of discourse. Either view is possible and can be presented systematically.

It is most important to note though that the two alternative views above cannot be indiscriminately mixed in the same discussion without leading to confusion, paradox, and error. It is unfortunate but true that many variants of modelling approaches, both in practice and as described in the literature, suffer from precisely these problems.

The current work of WG3 is based on the assumption that the conceptual schema and the information base should describe the conceptual view. This implies that the conceptual schema is defined in terms and constructs referring to things in the universe of discourse itself and expressing states of affairs about those things.

The constructs used in a conceptual schema and information base should be based on the fundamentals of formal logic as theoretical foundation. However, how closely they must follow the spirit and notation forms of these fundamentals is a subject of further investigation. It is quite possible to limit them to elementary constructs expressing these fundamentals. However, it will be

always possible to define, upon these fundamentals, a variety of more complex constructs ("macro constructs"), that may be more convenient or efficient for describing various aspects of a universe of discourse.

The choice of specific macro constructs is based on practical arguments such as ease of understanding and use. That choice is considered to be dependent on the application area of the information system for which a conceptual schema and information base has to be provided.

As already stated, it is important to distinguish carefully between the entities and their descriptions. In these descriptions entities are usually identified by names that refer to those entities. This includes synonyms - different names refering to the same entity - and homonyms - identical names refering to different entities. The relevance of this distinction, not only for information systems in particular, but for human communication in general, has been well-known in language philosophy and linguistics for a long time. Therefore the constructs should provide for synonyms and possibly cope with homonyms.

The conceptual schema not only describes the static aspects and dependencies of the universe of discourse, but also the dynamic aspects. This determines what manipulations of the descriptions are allowable as well as what descriptions may be present in the conceptual schema and information base. Therefore it may be clear that constructs have to be available both for the descriptions and for their manipulation in the information system.

The subject is elaborated further in chapter 3.

1.8. GUIDELINES FOR THE CONTENTS OF A CONCEPTUAL SCHEMA.

Since the selection of what are considered to be general classifications, rules, etc. of the universe of discourse is to a certain extent arbitrary, it follows that the choice of which should be described in the conceptual schema and which in the information base is arbitrary to a similar extent. In practice, however, the systems designer might consider various factors in deciding the boundary of the conceptual schema. These might include:

- describing classes (types, variables) in the universe of discourse rather than individuals (instances),
- describing concepts that are less subject to change rather than concepts that are changing more frequently,
- inclusion of rules or constraints having wide influence on the behaviour of the universe of discourse (and therefore on the behaviour of the conceptual schema and information base) rather than narrow influence.

At all times the following general principles for the conceptual schema should be observed:

* 100 Percent principle:
  All relevant general static and dynamic aspects, i.e. all rules, laws, etc., of the universe of discourse should be described in the conceptual schema. The information system

cannot be held responsible for not meeting those described elsewhere, including in particular those in application programs.

* Conceptualization principle:
A conceptual schema should only include conceptually relevant aspects, both static and dynamic, of the universe of discourse, thus excluding all aspects of (external or internal) data representation, physical data organization and access as well as all aspects of particular external user representation such as message formats, data structures, etc.

A more detailed discussion may be found in chapter 3.

1.9. ROLES FOR A CONCEPTUAL SCHEMA.

A fundamental impact of a conceptual schema is that the concepts used harmonize - and to a certain level make possible - human communication. Moreover, these concepts will influence the methods and results of analysing organizations and their information needs. In a way, a conceptual schema constitutes a general agreement concerning how to perceive a universe of discourse. This agreement may alter over time, but supports the evolution of applications over their life cycles as well as changes of this agreement itself (cf. The Helsinki Principle).

A conceptual schema is intended to properly describe the behaviour of a universe of discourse. Therefore the rules given therein, naturally, restrict possible evolutions and manipulations of the description of the universe of discourse, i.e. of the conceptual schema itself as well as of the information base.

Mainly for reasons of human convenience and efficiency, different users within the environment of a common information system will use different forms of external data representing the information. At the same time, for reasons of machine and storage handling efficiency, internal data organizations will be designed and used that may or may not differ from those external forms. In this context, the conceptual schema enforces preservation of meaning in transformations between the various data representations and defines the interpretation of these representations.

Therefore, and considering what is outlined in the previous sections, the following fundamental roles for a conceptual schema have been identified:

1. To provide a common basis for understanding the general behaviour of the universe of discourse;

2. To define the allowed evolution and manipulation of the information about the universe of discourse;

3. To provide a basis for interpretation of external and internal syntactical forms which represent the information about the universe of discourse;

4. To provide a basis of mappings between and among external and internal schemata.

1.10. REQUIREMENTS FOR A CONCEPTUAL SCHEMA FACILITY.

It is anticipated that the data base management systems developed in the future will include a component for handling a conceptual schema definition which will fulfil the roles mentioned in the previous section. In the course of time, provision of such a component should become a standard requirement.

To fulfil the roles indicated above, a conceptual schema facility must satisfy the following requirements:

1. It must provide basic concepts which are suitable for adequately describing both the static and dynamic aspects of a universe of discourse and ipso facto its description in terms of a conceptual schema and information base.

2. It must provide a language in which a conceptual schema can be expressed so as to be readily understandable to a user of the facility.

3. It must provide a language for precisely communicating a conceptual schema to a computer.

4. It should provide for easily modifying the conceptual schema to reflect changes in the general classifications, rules, laws, etc. of the universe of discourse, and for predicting the direct consequences of such changes.

5. The views of the information that different users wish to see are limited to those which do not contradict the assertions in the conceptual schema. If such external schemata are subject to change, the facility should be such that this should not affect the conceptual schema.

6. The conceptual schema should be kept invariant by the facility with respect to changes in the internal (physical) representation of the data within a computer.

The two languages mentioned in 2 and 3 may be the same but are not necessarily so. For the former purpose an additional graphic notation may be helpful.

These six requirements as a minimum must be met by any candidate conceptual schema facility.

1.11. REFERENCES.

[1] ANSI/X3/SPARC, ´Study Group on Data Base Management Systems: Interim Report 75-02-08`, In: ACM SIGMOD Newsletter, FDT, Vol 7, No. 2, 1975.

[2] TSICHRITZIS, D., and KLUG, A. (eds.) ´The ANSI/X3/SPARC DBMS Framework. Report of Study Group on Data Base Management Systems`, AFIPS Press, Montvale NJ, 1977.

[3] BSI/DPS13/WG1, ´Role and Composition of a Conceptual Schema in an Overall DBMS Architecture`, 1976.

[4] AFNOR/Z6/SC5, ´Study of Users Requirements`, 1977.

[5] NIJSSEN, G.M. ´The Next Five Years in Data Base Technology`, In: Infotech State of the Art Conference, London, 1977.

[6] NIJSSEN, G.M. ´A framework for advanced mass storage applications`, In: Medinfo 80, Proceedings of the Third World Conference on Medical Informatics, Tokyo 1980, North-Holland Publishing Company, 1980.

[7] FALKENBERG, E. ´On the conceptual approach to data bases` In: International Conference on Data Bases, Aberdeen 1980, Heyden and Son, London, 1980.

This page intentionally left blank

CHAPTER 2. FUNDAMENTALS FOR A CONCEPTUAL SCHEMA AND AN INFORMATION BASE.

2.1. GENERAL CONCEPTS AND DEFINITIONS.

We propose in this Report concepts and terms to be generally used in designing, describing, and using conceptual schemata and information bases. Some of the terms defined in this Report are already found throughout the data base literature, occasionally with conflicting meaning. The purpose of this chapter is to describe the fundamentals and to give short, precise, although intentionally informal, definitions of the concepts and terms, wherever possible conforming with the meaning most closely associated with natural language [1]; e.g. the term "real world" is to be interpreted in ordinary language sense. The definitions are embedded in explanatory prose to inform the reader of the basic concepts and intent of our view of the conceptual schema and information base. The definitions themselves will be summarized in a glossary of terms in appendix A to this Report.

We start by assuming that it is possible to select a portion of a real or hypothetical world that is describable in some chosen precise and formally defined language. All things we perceive or assume to exist in this selected portion of a world are called entities:

> ENTITY
> Any concrete or abstract thing of interest, including associations among things.

For example, if we select a portion of a world as described in appendix B and in which a certain Registration Authority is assumed to be interested, then entities are the car Ford Mustang PCXX999, the person Mr. Johnson, the date 29 January 1975, etc. A particular example of an abstract entity is an association among other entities, e.g. the "ownership" of the car PCXX999 by Mr. Johnson.

In perceiving or imagining the selected portion of a world we are interested in we may conceive all kinds of states of affairs concerning one or more entities therein. Examples are:

- The car PCXX999 is of model Mustang
- The car PCXX999 has got registrationnumber GMF 117
- The car PCXX999 is distributed by Ford to Smith's garage on 29 January 1975
- Garages sell cars to persons
- Ford is a car manufacturer

We call such states of affairs propositions:

PROPOSITION
A conceivable state of affairs concerning entities about which it is possible to assert or deny that such a state of affairs holds for those entities.

A proposition can concern one entity, several individual entities, groups of entities, etc.

In practice the distinction is often made between propositions about the actual state of individual entities, and propositions about which behaviour of entities may or may not be permissible or possible. The words "rule" and "constraint" refer in particular to propositions of this latter kind.

Actually it will be descriptions of the propositions - sentences - that enable us to discuss entities and their states of affairs at all - that is, to exchange information about entities by describing propositions which hold for them:

SENTENCE
A linguistic object which expresses a proposition.

LINGUISTIC OBJECT
A grammatically allowable construct in a language.

Note, that linguistic objects may be considered entities.

Sentences consist of terms and predicates:

TERM
A linguistic object that refers to an entity.

PREDICATE
A linguistic object, analogous to a verb, which says something about an entity or entities to which term(s) in the sentence refer.

For instance the sentence:

"The car PCXX999 is of model Mustang."

expresses the first example proposition above. In this sentence the verb "is of" formulates the predicate. The terms "the car PCXX999" and "model Mustang" refer to the involved entities.

Often various sentences convey the same information, and, in particular, different terms may refer to the same entity. For example, the term "Mary Jones" is evidently different from the term "Mary Smith". Nevertheless, after Mary Jones has married John Smith, both terms will be associated with the very same girl. Thus, the sentences

"Mary Jones was born in 1955"

and

"Mary Smith was born in 1955"

have the same meaning, and it is evidently the girl who was born in 1955, not the term! The notion of similar information conveyed by different sentences, in many cases resulting from alternate ways to refer to entities, has tremendous importance for flexible and unambiguous communication.

Some terms are simple linguistic objects, as for instance the terms in the above examples. In other cases, however, more complex linguistic objects may be used as terms. In the sentences

"Ford produces the car PCXX999."

and

"The manufacturer that produced the car PCXX999 distributes the car PCXX999 to Smith's garage."

the terms "Ford" and "the manufacturer that produced the car PCXX999" are two terms that refer to the same entity. In this example the first term is only referring to the Ford company. The second term also refers in a certain way to a proposition about the production of a car.

Some linguistic objects play no other role in the descriptions than to be used as names for something else. We will call them lexical objects or names:

LEXICAL OBJECT or NAME
A (simple) linguistic object that is used only to refer to an entity.

In normal cases a lexical object consists solely of one or more nouns.

The special kind of association between the "basic" entities and the lexical objects that refer to them we could call a naming convention. When such a naming convention between an entity and a lexical object is correctly established it is always possible, at least in principle, to identify a causal chain to the use of that lexical object from an instance of "name giving" in the world, i.e. a point in space and time where an appropriate action was taken that asserted, in effect: "Henceforth this entity will be called by the name (i.e. lexical object) so-and-so !".

It is part of the information system designer's job to make sure that all entities of interest can be referred to in some way. For this reason, the information system designer will generally wish to additionally describe, in the conceptual schema and information base, the commonly agreed ways to refer to entities.

It should be carefully noted, that there is no barrier and, indeed, often considerable utility in the same entity having more than one lexical object associated with it. These lexical objects then are synonyms:

SYNONYMS
Different terms that refer to the same entity.

Fundamentally there also is no barrier in several identical lexical objects being associated with different entities. These lexical objects then are homonyms:

HOMONYMS
Identical terms that refer to different entities.

In practice, they may cause some ambiguity. Therefore in some information systems homonyms are excluded. However, this exclusion is certainly not a fundamental or necessary requirement, provided some mechanism exists to resolve ambiguity.

Often we will be interested in entities that are considered to occur together:

ENTITY WORLD
A possible collection of entities that are perceived together.

For example, all cars, registered by the Registration Authority, and all manufacturers, garages and persons involved with those cars, as described in appendix B, may be considered to form an entity world.

Many different entity worlds can be discerned at the same or at different times. Also, an entity can belong to many entity worlds.

A collection of propositions asserted to hold for a given entity world is called a proposition world:

PROPOSITION WORLD
A collection of propositions each of which holds for a given entity world.

A collection of sentences that express the propositions of a proposition world informs us about the relevant entity world.

Our selected portion of a real or hypothetical world involves all possible entities we are interested in. These are the ones we may want to discuss:

UNIVERSE OF DISCOURSE
All those entities of interest that have been, are, or ever might be.

The universe of discourse might alternatively be called the universe of possible entities. Note, that the universe of discourse is limited to the possible entities we are interested in and therefore want to discuss or describe. Taking the example of appendix B the universe of discourse of the Registration Authority consists of all cars, manufacturers, car models, garages, persons, etc., that have existed, exist, or ever might exist and in which the Registration Authority will be interested.

All propositions that may hold in any one or more entity worlds that together constitute the universe of discourse, form the universe of possible proposition. However, for an information system designer not all of those propositions are of prime interest. What he is looking for in the first place are those propositions that hold for all possible entity worlds:

NECESSARY PROPOSITION
A proposition asserted to hold for all entity worlds and therefore must be part of all possible proposition worlds.

Since necessary propositions are states of affairs that necessarily hold for the involved entities in all entity worlds, they often have a more "general" character. These necessary propositions form an abstraction of all entity worlds, generalizing what they have in common.

Necessary propositions define which entities may occur in any entity world - possible entities, and in relevant cases, which entities must occur in each entity world - necessary entities.

The classifications, rules, laws, etc., of the universe of discourse, which are mentioned in chapter 1, constitute the necessary propositions. The section B.2 of appendix B describes informally the necessary propositions of our example universe of discourse.

Some necessary propositions that hold for each and every registered car in all entity worlds containing registered cars in our example universe of discourse are:

> "A car is of a particular model."
>
> "Each car has a registration number given by the Registration Authority at the time the car is registered."

However, we do not wish to limit the necessary propositions to only general states of affairs. States of affairs involving one or a few particular entities can necessarily hold for all entity worlds. E.g.:

> "Only 5 manufacturers can have permission to operate in the same period of time."
>
> "Fuel consumption is between 4 and 25 litres per 100 kilometres."

As already stated, necessary propositions tend to have a more general character, that is, they hold for collections of similar entities - classes of entities:

> CLASS (of entities)
> All possible entities in the universe of discourse for which a given proposition holds.

Each class of entities is determined exactly by its possible members. Clearly any particular entity may be a member of many classes, so that classes in general are not disjoint.

The proposition that determines the class might be a state of affairs of arbitrary complexity. E.g.:

- The class of Car Manufacturers consists of all possible entities that produce a car.
- The class of Car Owners consists of all possible entities that either belong to the class of Car Manufacturers, Garages, or Persons, and that own a car.

Classes themselves are entities, and, as the examples already show, can be

given names.

The general notion expressed in information processing literature as "type" is that of "class" or more precisely "class-membership":

> TYPE (of an entity)
> The proposition establishing that an entity is a member of a particular class of entities, implying as well that there is such a class of entities.

In other words the sentences

> "The entity x is a Car Manufacturer (type)"

and

> "The entity x belongs to the Car Manufacturers (class)"

convey exactly the same information.

A type can be referred to by means of a type-name. Quite often a singular form of a name (noun) in such cases is used as type-name, while the plural form is used as class-name.

Whether a type notion will be associated with a particular class of entities is an arbitrary choice of the information system designer, often inspired by what is considered practical or usual in the user's environment of the conceptual schema.

The notion "instance" or "occurrence" is usually associated with the notion of type:

> INSTANCE or OCCURRENCE (of an entity-type)
> An individual entity, for which a particular type proposition holds, that is, which belongs to a particular class of entities.

In designing information systems the notions of class and type are used in particular to establish collections of necessary propositions: With a specific class or type, a collection of relevant necessary propositions may be identified, that hold for all possible entities which are members of that specific class. E.g. in the example of the Registration Authority the following necessary propositions hold for all possible entities that are cars:

- a car is produced by a car manufacturer
- a car has a serial number
- a car is of a car model
- a car is given a registration number by the Registration Authority

etc.

The propositions that determine such classes or types belong themselves, of course, to the necessary propositions in these cases.

The formal description of the necessary propositions is called the conceptual schema:

CONCEPTUAL SCHEMA
A consistent collection of sentences expressing the necessary propositions that hold for a universe of discourse.

It follows from the above that all possible entity worlds constituting the relevant universe of discourse share a conceptual schema. This conceptual schema in fact establishes the universe of discourse as it informs us what exactly the collection of all possible entities may be.

What propositions are necessary propositions, and therefore what the boundaries of the conceptual schema will be, is arbitrary, and depends on how detailed the information system designer wishes to be. Moreover, this may change over time requiring additional changes to an already formulated conceptual schema.

Propositions may hold in a specific entity world in addition to the necessary ones formulated in the conceptual schema. The description of those additional propositions constitutes an information base:

INFORMATION BASE
A collection of sentences, consistent with each other and with the conceptual schema, expressing the propositions other than the necessary propositions that hold for a specific entity world.

Note, however, that the collection of sentences constituting the one conceptual schema and a specific information base together describe all propositions considered relevant for a specific entity world and therefore describe a specific proposition world for that entity world. These propositions are conceived to hold for the entity world, the latter being perceived as "reality". For that reason, this collection of sentences constituting the conceptual schema and information base must necessarily be consistent, if it purports to be a truthful description of those propositions.

Actually it is the information base together with the conceptual schema that in essence establishes a particular entity world. In other words the entity world consists exactly of those concrete or abstract objects - entities - that are referred to by the terms in the sentences contained in the information base and conceptual schema together.

Note, that it may very well be possible to describe one universe of discourse or one particular entity world in more than one conceptual schema and information base. We assume, however, that usually only one conceptual schema and one information base will be part of one information system at a time.

Often, but not necessarily always, an information base is meant to inform us about the entities that occur in the instant or period of time, usually referred to as "now". A "current" state of an information base - an actual information base - however, may refer to a "past" or "future" entity world:

ACTUAL INFORMATION BASE
That information base which exists in a specified instant or a period of time, usually referred to as "now", and which expresses the additional propositions other than the necessary ones, that hold for an entity world.

ACTUAL ENTITY WORLD
A collection of entities of interest that is described in an actual information base and its conceptual schema.

The entity world described in sections B.2 and B.3 of appendix B could be considered as the actual entity world of interest to the Registration Authority, covering at least the period of time from 1975 until today.

2.2. BASIC CONCEPTS AND DEFINITIONS FOR ACTIONS ON THE CONCEPTUAL SCHEMA AND INFORMATION BASE.

The information base and the conceptual schema will change with time in order to reflect changes in the selected portion of a world constituting the universe of discourse, since only sentences asserted to be true of the universe of discourse should be in the information base or conceptual schema.

Changes in the selected portion of a world are for example:

* Entities appearing or disappearing in the selected portion,
* An entity changing its state of affairs or associations with other entities,
* The classification of entities or some rules or constraints about entities changing,
* The scope of interest changing, so that the selected portion itself expands or shrinks.

All such changes may require changes to both the information base and the conceptual schema. Although the first two kinds of changes might limit resulting changes to the information base only, the latter two kinds mentioned will certainly cause changes to the conceptual schema as well.

The basic notion of information manipulation in an information base or conceptual schema is an elementary action. Three kinds of elementary actions are defined: insertion, deletion, and retrieval.

INSERTION
The addition of a sentence to the information base or conceptual schema. Other sentences, not deducible before insertion may become deducible and therefore become a deducible part of the information base or conceptual schema.

Note, that a deducible sentence will not automatically be actually inserted.

A typical example may be the insertion of the sentence:

"On 29 January 1975, the car GMF117 is distributed to Smith's garage".

From this it might be assumed deducible that from 29 January 1975 onward:

"Smith's garage owns car GMF117."

The next elementary action is defined as:

> DELETION
> The removal of a previously inserted sentence from the information base or conceptual schema. Any deducible sentence, which cannot be deduced without the deleted sentence, will no longer be deducible and therefore no longer be a deducible part of the information base or conceptual schema.

Note, that a deducible sentence may have been actually inserted. In that case, the deletion of another sentence, on which the deducibility of the sentence essentially depends, will not automatically delete that actually inserted sentence. The deletion of that other sentence only makes impossible the deduction of the actually inserted sentence.

If we consider the example of an insertion mentioned above, and assume that the deducible sentence

"Smith's garage owns car GMF117"

has been actually inserted, then the deletion of the sentence

"On 29 January 1975 car GMF117 is distributed to Smith's garage"

will not result in the deletion of the sentence stating who owns the car, but we can no longer deduce that sentence.

The last elementary action is defined as:

> RETRIEVAL
> To make known a sentence which has been inserted in the information base or conceptual schema, or is deducible from other sentences in the information base or conceptual schema.

Note, that the retrieval of a deducible sentence from the information base or conceptual schema is possible only if the information system knows how to deduce this sentence from other available or deducible sentences in the conceptual schema and information base.

Combinations of elementary actions intended to achieve a specific result may be allowed. Such combinations are defined as:

> ACTION
> One or more elementary actions that, as a unit, change a collection of sentences into another collection of sentences in the information base or conceptual schema and/or make known a collection of sentences present in the information base or conceptual schema.

A typical example of an action is the replacement of a particular sentence by another one, i.e. a deletion followed by an insertion. Due to the many cases where this particular class of actions is applicable, it seems to be convenient to formulate this special kind of action as:

MODIFICATION
The replacement of a sentence in the information base or conceptual schema by another one, thereby possibly changing the collection of sentences which are deducible.

To control actions and rule out impermissible ones, it will be necessary to impose rules or constraints on actions. Therefore the following definition is added to define actions that are considered to be atomic "execution units":

PERMISSIBLE ACTION
An action, conforming to specified rules or constraints, which
- changes a presumably consistent collection of sentences in the information base or conceptual schema into a consistent collection of sentences

and/or
- makes known a consistent collection of sentences present in the information base or conceptual schema.

We should point out here that only the final collection of sentences as a result of a permissible action need be a consistent collection of sentences. Intermediate collections, if recognizable, need not be consistent.

Note, that certain permissible actions may change a presumably consistent, but actually not "truthful" collection of sentences into a consistent and truthful one. Such permissible actions will be needed to correct corrupted information bases or conceptual schemata, whatever the reason of the corruption may be. These specific permissible actions may be allowed to ignore certain rules about permissible or required sequences of state of the involved collections of sentences. For example, if it is erroneously stated that a person is married then changing this information to the statement that that person still is single may involve such a special permissible action (cf. the examples in section 2.7).

An elementary action is caused by an elementary command to the information system:

ELEMENTARY COMMAND
The order or trigger for an elementary action to take place.

Both an action and a permissible action are caused by a command to the information system:

COMMAND
The order or trigger for an action or permissible action to take place.

However, as response to the command, the action may be refused if the permissibility would be violated.

Commands and actions must be described in a suitable language:

COMMAND STATEMENT
A linguistic object expressing a command or elementary command.

In other words, presenting a command-statement to the information system constitutes the command.

It will also be necessary to have facilities designed to express the combination of elementary actions and the identification of them as one unit defining a single action or permissible action:

> ACTION DESCRIPTION
> A linguistic object describing an action or permissible action.

The possible syntactic and semantic complexity of command statements and action descriptions (e.g. structure of description and expressive power in terms of what commands and actions are describable) depends on the language chosen.

2.3. THE BEHAVIOUR OF AN INFORMATION PROCESSOR.

As already mentioned in chapter 1, the interaction between environment and information system takes place by means of messages:

> MESSAGE
> A collection of one or more sentences and/or command statements to be used as an information exchange between the environment and the information system.

Messages are dealt with by the information processor part of the information system:

> INFORMATION PROCESSOR
> The mechanism that in response to a command executes an action on the conceptual schema and information base.

The information processor recognizes whether or not messages received from the environment belong to a given language. Messages which do not belong to this language are discarded as irrelevant. Valid messages may express a change in the universe of discourse, or require to make known one or more sentences present in the conceptual schema, or in the information base, or deducible from sentences present in them.

A message expressing a change in the universe of discourse must contain or be accompanied by a command statement identifying the action description - or the action description itself - for a permissible action to effect the appropriate change in the conceptual schema and information base. The information processor interprets the command and changes the information base or conceptual schema according to behaviour rules or constraints. These rules not only determine whether the resulting collection of sentences in the conceptual schema and information base will be consistent, but also decide whether the conceptual schema and information base may be changed at all depending on what is already present in the conceptual schema and information base.

The information processor will refuse the command if the change cannot be effected according to the behaviour rules or constraints. As a result of a refusal, the information base and the conceptual schema will be as if the command had never been issued (i.e. completely unchanged).

If a message expresses a command to make known a collection of sentences present in the conceptual schema and information base or deducible from them, the information processor interprets the command. It issues a message reporting the appropriate collection of sentences according to behaviour rules or constraints which specify when and which collection of sentences present in the conceptual schema and information base shall be reported as a result of the command. The rules include the inference rules in case deducible sentences are involved.

The information processor will refuse the command if any behaviour rules or constraints would be violated in reporting the required sentences.

When it issues a command, the environment needs to know if the command is honoured or refused. Therefore the information processor must issue the environment a message to that effect.

In other words, the result of a permissible action is a function of "controlling" sentences stating rules and constraints, the sentence(s) to be changed or made known, and the incoming message including the command statement and additional sentences if relevant.

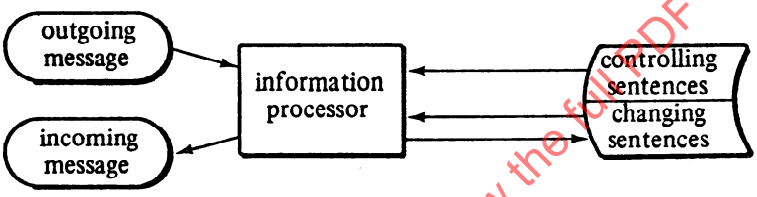


Figure 2.1. The information processor in action.

Usually the controlling sentences are largely in the conceptual schema although sentences in the information base as well can have a controlling role. The sentences to be changed or made known, are in normal application cases, all in the information base. However, the involved sentences are in the conceptual schema in cases where the conceptual schema itself is subject to change or reporting. Note, that in this latter case some of the controlling sentences may be found in the information base also.

A sentence expressing a proposition as such has a truth value. The following principle is axiomatic:

> The truth value of a sentence is considered true if it is explicitly stated true by a message accepted from the environment according to the rules known to the information processor. It is also considered true if it is deducible from such explicitly stated sentences in accordance with the inference rules known to the information processor.
>
> The truth value of any sentence, whose denial can be deduced similarly from explicitly stated sentences, is considered to be false.
>
> The truth value of all other sentences is considered to be undecidable.

The task of the information processor is twofold. Sometimes the emphasis lies on deciding whether changing a collection of sentences is permissible or not; at other times the deduction of "new" sentences or the generation of outgoing messages seems to be more important. Therefore the rules or constraints controlling the first role sometimes are called permissive rules or constraints; the rules prescribing the sequence of actions, to be performed including the generation of deducible sentences and messages, are referred to by the term prescriptive rules or constraints.

2.4. INSERTING A CONCEPTUAL SCHEMA - THE MINIMAL CONCEPTUAL SCHEMA.

Initially, the only rules known to the information processor are those built into the information processor itself. This basic set of rules provides the information processor with an interpreter mechanism that enables the information processor to at least recognize and interpret a minimal language. This language allows the environment:

* to extend the language to include constructs appropriate for the description of the universe of discourse,
* to specify commands,
* to specify authorizations,
* to specify new behaviour and inference (deduction) rules relevant to the conceptual schema and information base,
* to introduce action descriptions for permissible actions.

Further, the information processor will be equipped with a set of algorithms allowing it to derive new sentences from the already available ones.

These built-in behaviour rules constitute the minimal conceptual schema of the information system.

Given this built-in minimal conceptual schema and an initially empty information base it is possible for the environment to use this minimal language to build up the required conceptual schema as well as the information base in a systematic fashion.

There must be a fundamental rule concerning the proposed insertion of a new rule or constraint in order to decide what to do in the event that this insertion would make certain collections of sentences, already present in the conceptual schema and information base, no longer consistent. Such a fundamental rule can reject the existing sentences or the proposed new rule. It must not accept both.

At the beginning any sentence accepted by the initial built-in behaviour rules probably becomes an additional rule or constraint by being included in the conceptual schema or possibly in the information base. As sentences are added, the number of rules or constraints specified in the conceptual schema and information base increases and so further constrain and control the permissible actions to the conceptual schema and information base.

2.5. BEHAVIOUR RULES FOR THE ENVIRONMENT.

The behaviour rules mentioned so far establish the behaviour of the information processor and what is permissible in the information base or conceptual schema. A distinction has been made between permissive rules and prescriptive rules.

Information systems may issue messages to the environment which are intended to cause change in the environment. The behaviour rules within such information systems must be extended accordingly. This situation, however, does not imply that the information system controls the environment. Firstly these messages are generated according to rules or constraints established by the environment. Secondly, the information system cannot force the environment to obey rules expressed by such messages.

The additional rules consist not only of permissive and prescriptive rules for the information processor, but also of permissive and prescriptive rules for the environment. The permissive rules for the environment establish the criteria needed for the information processor to test the actual information, so that the information system can generate warning messages. The prescriptive rules for the environment form the base for the information processor to generate appropriate requests to the environment.

These latter behaviour rules for the environment will never dictate what is permissible in the information base or conceptual schema including consistency rules for collections of sentences. As far as the information system is concerned, they formulate only what is desirable. As such they form a separate class of rules or constraints in the information system. Note, that a undesirable collection of sentences will nevertheless be consistent.

For example, if a behaviour rule of the universe of discourse prescribes that only black Ford cars should be produced, then the reporting of a red Ford car being produced must not be rejected by the information processor; instead a warning message should be issued to the appropriate authorities in the environment.

Summarizing we may say:

- What is considered to be impossible in the universe of discourse or environment, establishes what is not permissible in the information system including whether a collection of sentences is consistent (behaviour of the information system).

- What is considered to be not permissible in the universe of discourse or environment, is undesirable for the information system, although it is perfectly permissible (behaviour of the environment).

2.6. STATIC AND DYNAMIC RULES AND CONSTRAINTS.

Static aspects of a system are those which apply to each of its individual states. Static rules or constraints establish dependencies between parts of the system at any one instant of time. Dynamic aspects are those which govern the evolution of the system. To discuss dynamic aspects of a system means to dis-

cuss its laws of change. Dynamic rules establish dependencies between parts of the system through several instants of time.

The system we are considering here is the information system together with its environment and the universe of discourse. Rules and constraints deal with dependencies within and between all of these. In this generality, they touch on some subtle problems, in particular when dynamic aspects are involved. For a thorough treatment they require more elaboration of concepts. Therefore, we will here outline a few aspects of dynamic rules in the general sense, and then restrict ourselves to static and dynamic rules in a more special sense.

The causal structure of interdependencies is an important aspect of dynamics in the general sense. A typical example of such dependency concerns co-ordination of permissible actions: There may be rules requiring certain conditions to hold before specific actions can take place (see section 2.8). Another typical issue is to describe what messages cause the information processor to perform which changes to the information base or conceptual schema and to return which answers, say, results of retrievals or reporting on actions (cf. sections 2.2 and 2.3).

A third important example is the subject of authorization. In general, there must be authorization rules that control whether a user is entitled at all to give a command for a particular permissible action changing or retrieving a particular collection of sentences, or is entitled to receive a particular message from the information system. This implies that identification of the source and destination of messages and commands are involved in the enforcement of authorization rules. The subject is not yet discussed further in this Report.

Although a comprehensive information system design must be aware of all above mentioned relevant aspects, the focal points of interest traditionally are the rules and constraints for the sentences administered by the information processor. Therefore, and because of the complexity of the general treatment, we shall consider in this section the rules or constraints in the special sense of permissibility of information base states - that is, collections of sentences - and sequences of information base states.

Static rules and constraints under this restricted view are then concerned with the consistency and permissibility of collections of sentences. The effect of a static rule may be locally restricted to single sentences or it may globally involve several sentences within the same collection of sentences in one state of the information base. An example of a locally effective rule is the requirement that the serial number of a car must be, say, a natural number less than 10 000 000. A rule that requires the serial numbers of cars to be unique has a very broad global effect. To be enforced or checked, it needs a complete survey of all instances of serial numbers of cars registered in the information base. Static rules may also be of very different complexity. Examples of more complex conditions are functional dependency or set inclusion. Detailed examples of static rules may be found in the appendices D, E, and F of this Report.

Dynamic rules under the restricted view in this section are concerned with the permissible transitions from one collection of sentences to a next one and thus specify the possible sequences of information base states. Therefore they will be called transition rules to distinguish them from the dynamic rules in the general sense. Transition rules abstract from causes for changes as well as from effects the changes may have on the environment (e.g. reactions to retrieval results or to triggered messages from the information processor to the en-

vironment). They simply describe which information base states may occur subsequent to other given information base states.

Satisfaction or violation of transition rules can sometimes be checked by inspection of states. In fact, static rules can be regarded as special cases of transition rules. Although static rules define which states are permitted, a static rule can be re-interpreted as stating that certain states are permitted or forbidden, no matter what the previous state was. This can, however, also be expressed as a transition rule. The "no matter what the previous state was" can be taken care of by admitting all information base states as possible most recent information base states.

It may be argued that permissibility of an information base state may depend on any of the former states, rather than just on the most recent one. However, as an additional postulate, it is assumed that the history of states can affect permissibility only in as much as the history is reflected in the most recent state. It is therefore sufficient to take into account the most recent state only (cf. "actual information base" in section 2.1).

Some special attention must be given to the kinds of rules that are involved in derivations of deducible sentences. For example, the fuel consumption rate of a car may be given as the fuel consumption per 100 kilometres. That is, given a sentence stating the kilometrage of a particular car and a sentence stating the amount of fuel consumed, the fuel consumption rate can be derived, provided a sentence stating the definition - rule - for the fuel consumption rate is also available.

Actually, such rules are static rules as they deal with sentences in one information base state. Some people, however, may consider them dynamic rules, as they "control" possible derivation processes. Several authors use the term "derivation rule" for such rules.

Note, that deducible sentences need not only be derived by derivation processes within the information system. It is quite possible that such sentences are explicitly inserted. In such cases these "derivation rules" control the consistency of the resulting collection of sentences including the inserted "derivable" sentence.

The above mentioned rules not only include those needed for what is commonly considered as derivable information. The decision rules needed in automatic process control systems and decision support systems are also in this category. Therefore, they belong to the necessary propositions in most cases and are thus an essential part of the rules described in a conceptual schema.

2.7. EXPRESSING RULES AND CONSTRAINTS.

We will consider two different ways of specifying transition rules, the state-oriented and the action-oriented descriptions.

With the state-oriented descriptions, the rules and constraints are given as requirements on subsequent information base states. A rule or constraint is then basically a description of a set of pairs of information base states <OLD, NEW>. A change from a state OLD to a state NEW is permissible - regardless of how it is effected - if and only if the pair <OLD, NEW> is in the set.

Any transition rule distinguishes permitted pairs of information base states from forbidden ones. Thus, it can be viewed as a binary-valued function which tags each pair of information base states with either "+" (permitted) or "-" (forbidden). This defines a dichotomy on the Cartesian product S X S of the set S of all information base states with itself. Such a dichotomy is a decomposition of S X S into two sets, T+ of permitted and T- of forbidden changes. T+ could be called the positive, and T- the negative extension of the rule. Either of the two sets can be used to describe the dichotomy. The effect of a transition rule is therefore completely captured by either its positive or its negative extension.

With the action-oriented descriptions, the permissible changes are given by admissible action sequences. The permissibility of an action or action sequence may depend on the present state. A rule or constraint is therefore basically a set of pairs which each consist of an information base state component and an action sequence component.

With the action-oriented descriptions, a rule specifies that a transition is permissible if, starting from a permissible state OLD, the transition is effected by an action sequence Q such that the pair <OLD, Q> is in the set described by the rule. Ultimately, permissibility may be traced back to an initial state and all actions performed on it until the present.

To be able to make finite descriptions of virtually infinite sets of action sequences - there is no restriction on the length of action sequences - it is necessary to define classes of actions. An action-oriented rule then refers to classes of action sequences. Conceivably, an action sequence may consist of only one elementary action (for "elementary action" and "action" see section 2.2). Complex rules are formed by composing actions to constitute a permissible action (for given departure states).

A permissible action succeeds or fails as a whole. The actions of which a permissible action is composed might not be permissible individually. Thus, given the rule that an employee must have a salary and must work for a department,

(INSERT "John works for Department Sales",
INSERT "John earns a salary of 20000")

may be permissible, while each individual INSERT action alone would violate the above mentioned rule. Permissibility in this case is dependent on both elementary actions happening together.

To demonstrate examples of static and dynamic rules let us assume that the marital state of persons might be defined as one of the following states: "single", "married", "widowed", or "divorced".

A static rule applicable on sentences within one collection of sentences may be:

> If a person is married to another person, then both persons must have a marital state of married.

The following table expresses an example set of state-oriented rules for permissible changes of marital state, defining a permissible sentence in the resulting collection of sentences as a function of a sentence in the initial collection of sentences:

from/to	SINGLE	MARRIED	WIDOWED	DIVORCED	UNKNOWN
SINGLE	-	yes	no	no	yes
MARRIED	no	-	yes	yes	yes
WIDOWED	no	yes	-	no	yes
DIVORCED	no	yes	no	-	yes
UNKNOWN	yes	yes	yes	yes	-

Figure 2.2. Transition matrix for marital state.

An example for an action-oriented rule is the following:

INSERT "x is married to y"
only if both x and y are not married.

These basic description possibilities are discussed in more detail, with some examples, in appendix G; appendices E and F also give some examples.

A suitable language is required for the specification of rules and constraints. The totality of established rules and constraints can be regarded as one comprehensive rule that controls the entire information base. Of course, in any reasonable language such an overall rule will not be given in one piece. Therefore, it is a requirement for the language to allow for composing complex rules and constraints from simpler ones. The decomposition into simpler rules must end with predefined primitive rules. A primitive rule would have to be a simply structured, easily surveyable set of pairs of information base states or of an information base state and an action sequence.

At present, no specific proposal is made for composition of rules and constraints, but a general aspect is considered. The rules may be expressed in a permissive or restrictive style. Composition may be additive or subtractive - that is, one rule may work in the same sense as, or counter to, another. The result may be a permissive or a restrictive rule. This gives a number of composition options, not all of which may be desirable for a practical specification language.

2.8. CO-ORDINATION OF PERMISSIBLE ACTIONS.

It should be noted, that the subject of this section in particular has only rather recently become a topic of research and discussion (e.g. [2, 3, 4, 5]). Therefore this section suggests more the directions in which development might go, rather than demonstrating and summarizing results.

All actions taken by the information processor on the information base and conceptual schema should occur in terms of permissible actions as defined in section 2.2. Because of this, any process performed by the information processor will consist of one or more permissible actions.

A permissible action is considered to be atomic and therefore uninterruptable. It is triggered by an appropriate command. The effect of a command might be a "chain" of permissible actions. That is, a permissible action may issue a command for other permissible actions.

The conceptual framework for co-ordinating permissible actions is based on the following concepts:

EVENT
The fact that something has happened in either the universe of discourse, or the environment, or in the information system;

COMMAND (as in section 2.2)
The order or trigger for an action or permissible action to take place;

PERMISSIBLE ACTION (as in section 2.2)
An action, conforming to specified rules or constraints, which
- changes a presumably consistent collection of sentences in the information base or conceptual schema into a consistent collection of sentences

and/or
- makes known a consistent collection of sentences present in the information base or conceptual schema;

COMMAND CONDITION
The precondition, including synchronization aspects, that must be met before a permissible action may take place.

The information system only reacts because of an event. The dependency between the event and the reaction could be perceived as in figure 2.3.

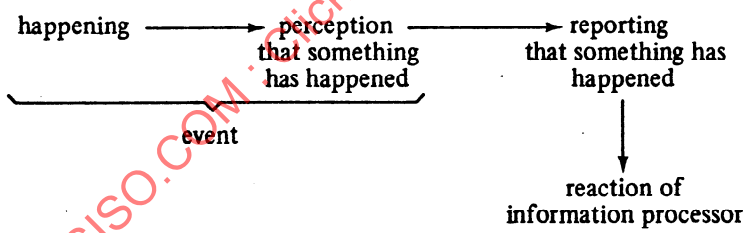


Figure 2.3. Dependency between event and reaction.

As far as the information system is concerned it is not relevant whether the event is the happening or the perception of the happening. However, the event (the stimulus) must cause a report of the event to the information processor, upon which the information processor has to react.

We define two kinds of events: external events and internal events.

EXTERNAL EVENT
An event that occurs in the environment or universe of discourse.

The reporting of such an event is a message to the information system expressing at least a command.

INTERNAL EVENT
An event that occurs because of the termination of some permissible action in the information sysem.

Depending on the result of such a permissible action, internal events are relevant only if the reporting of the termination must be followed by a succeeding reaction of the information system, or by a message sent to the environment.

Events have certain characteristics, which are important for the information system. In the first place, events may belong to certain types (classes) of event. Closely associated with the notion of event type is the notion of event occurrence.

An important characteristic of the event type, or rather its reporting, is the type(s) of command, associated with the event type, which determine the permissible action or permissible actions that are expected as a reaction of the information system.

Other important characteristics of the event types are:

- the number of event occurrences to be expected;
- the frequency of event occurrences.

Most events also need an associated reporting of certain "parameters" in order to direct the permissible action as to what precisely must be done. In those cases the message reporting the event also contains, or must be accompanied by, one or more sentences, called input sentences for the permissible action.

The permissible actions accomplished by the information processor may insert, retrieve, delete, or modify sentences in the conceptual schema and information base. It is not necessary, however, that input sentences for the permissible action be inserted themselves in the conceptual schema or information base, if the input sentence only provides parameters to the permissible action. Sometimes input sentences will be inserted, at other times they cause the insertion of other derived sentences. An input sentence which is not actually inserted, will generally not be reproducible after the termination of the permissible action.

A permissible action is considered as a "black box", that is, we are interested only in what the permissible action does - what its result is. This, however, may be specified in terms of (elementary) commands for (elementary) actions. Such specifications concentrate on the type of permissible action: The type of permissible action determines what the permissible action will do. This is described by the action description. The sentences involved in the permissible action establish the actual result. Together they establish the actual permissible action, that is, an instance of the permissible action type.

A permissible action is triggered by an event, or more precisely by the command expressed in the message reporting the event. The type of event determines what type of permissible action will be triggered. At that moment the permissible action becomes active. An active permissible action will be uninterruptable

until it is finished. This termination may raise an (internal) event, depending on the result of the permissible action.

At any time several permissible actions can be active in the information system. These permissible actions may belong to the same or different types of permissible actions. Two permissible actions which are active at the same time are considered to operate completely independently of each other.

As already indicated a certain delay of time may occur between the event itself and its reporting to the information processor, or between its reporting and the information processor's reaction. Therefore, it is possible that the sequence in which the events are reported to the information system may differ from the sequence in time of the events themselves occurring in the universe of discourse. If this sequence in time of events is relevant, for instance, because of necessarily preceding states of affairs that must have been recorded first in the conceptual schema and information base, the resulting permissible actions must be co-ordinated, including, if necessary, synchronization of the permissible actions.

The command condition of a certain permissible action consists of the (external and internal) events which must have occurred before triggering the permissible action, and a rule or set of rules which establish in what manner the events determine the necessary condition for the permissible action to be triggered. A command condition can be extended and made more precise by rules about sentences reporting the events or already present in the conceptual schema and information base.

If more than one event is involved in the command condition of a permissible action, then the last event occurring in time, independent of what event this may be, fulfils the command condition and the permissible action is triggered. The "arrival" of the other events except the last one brings the command condition to a "wait-state".

A command condition may have a limited wait-state, that is, the wait-state is only allowed to have a limited duration in time. If within this period the last event needed does not occur, then the permissible action will no longer be triggered because of the events that have already occurred. A new series of events, however, may prepare a new wait-state for a possible permissible action. When no limited wait-state has been defined, the wait-state may last eternally.

One event occurrence can only once take part in the triggering of one permissible action. When a command condition is met and the permissible action is triggered, then the "arrival" of new events can only prepare the command condition for a new triggering of the permissible action.

A special note within the context of command conditions is needed about the events. An event (occurrence) itself is information bearing, as it establishes that something has happened. This is not established in general terms, but in a most specific way. Therefore, an event does not establish that a happening of a certain type has occurred, but that that particular happening (occurrence) with that particular result took place. For example, a certain car PCXX999 has been produced by FORD Motor Company.

As a consequence, internal events are controlled by prescriptive rules for the event. A type of permissible action may possibly terminate in a number of dif-

ferent internal events depending on the result of the permissible action. Such various events may be mutually exclusive, but this is not necessary.

During the permissible action (and actually before the execution of the permissible action itself) the information processor deals with a number of rules and constraints:

1. It recognizes the reporting of the event and the reception of the input sentence according to recognition rules;

2. It evaluates the command condition and triggers the actual permissible action or keeps the command condition in a wait-state;

3. After triggering the permissible action it performs the permissible action itself according to prescriptive rules expressed in the action description of the permissible action manipulating the conceptual schema and information base;

4. It checks the manipulation of the conceptual schema and information base according to the appropriate static and dynamic rules and constraints for the conceptual schema and information base, and the appropriate authorization rules;

5. It possibly reports one or more appropriate internal events according to the prescriptive rules for the internal events;

6. It possibly generates an outgoing message according to prescriptive rules for the outgoing message.

In the following a graphic formalism will be used to discuss some possibilities of co-ordinating permissible actions. This formalism is derived from those used in Petri Network approaches [6]. In the diagrams the command condition mechanism (C) is depicted by a circle. The boxes depict permissible actions (PA). The incoming arrows show the sources of the events involved in the fulfillment of the command condition.

Co-ordination of permissible actions can involve all kinds of "chains" of permissible actions. Several possibilities exist, e.g.:

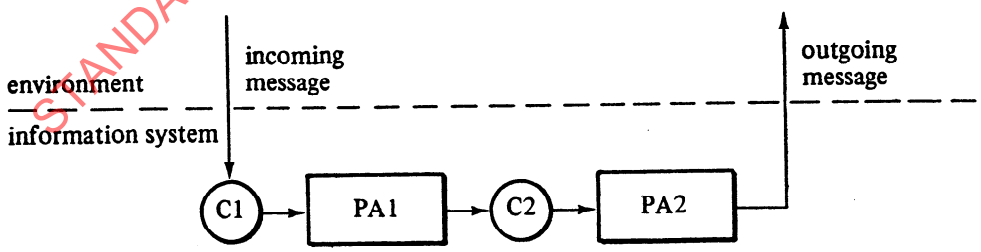


Figure 2.4. Simple sequence of permissible actions.

1. The result of an incoming message is an ordinary sequence of permissible actions. That is, a first permissible action takes place because

of the message from the environment. The arrival (external event) fulfils the command condition for this permissible action. The result of this permissible action (internal event) is the sufficient and only command condition to be fulfilled for a second permissible action. This is shown in figure 2.4.

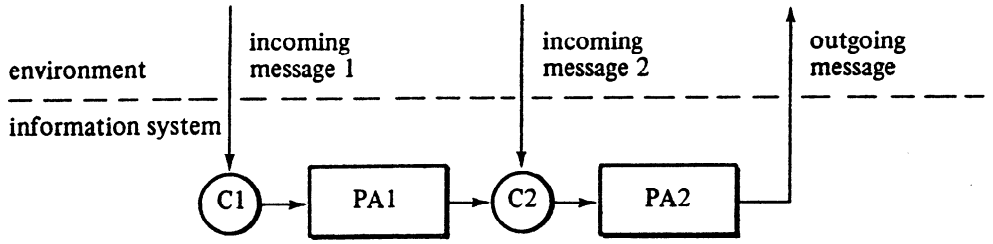


Figure 2.5. Conditional sequence of permissible actions.

2. The ultimate result of an incoming message is a sequence of permissible actions, provided additional incoming messages are received. In figure 2.5 the second permissible action will take place if both the first permissible action has taken place (internal event) and a second incoming message has arrived (external event). The one that occurs latest in time completes the command condition and therefore triggers the second permissible action.

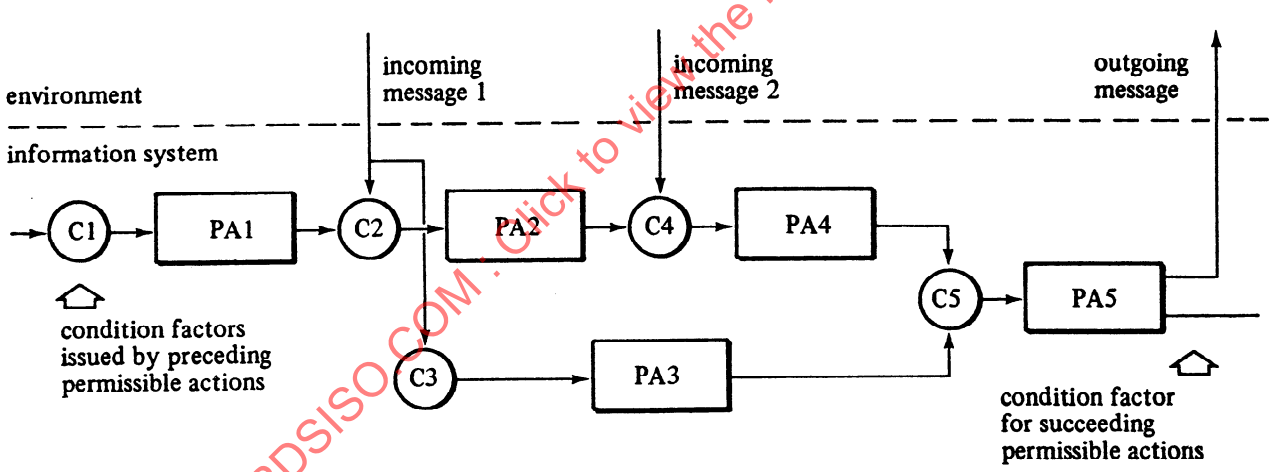


Figure 2.6. Co-ordinated sequences of permissible actions.

3. More complex situations are possible as for instance is depicted in figure 2.6. Note, that in this example permissible action 3 may be concurrent with the sequence of permissible action 2 and permissible action 4. Permissible action 3 and permissible action 4 both are predecessors to permissible action 5. Also note, that the arrival of incoming message 1 is part of the command conditions of both permissible actions 2 and 3.

Appendix H discusses some aspects of co-ordinating permissible actions in more detail and gives some demonstration examples.

2.9. REFERENCES.

[1] MURRAY, J.A.H. et al.
´The Oxford English Dictionary` with supplements, Clarendon Press, 1933 - 1977.

[2] HECKENROTH, H., TARDIEU, H., ESPINASSE, B.
´Modèles et outils pour la conception de la cinématique d´un système d´information`,
In: Rapport de recherche IRIA 310, CETE d´Aix, Grasce, Université d´Aix - Marseille, 1980.

[3] VALETTE, R., COURVOISIER, M.
´Recherche d´un modèle adapté au système de commande de processus à évolution parallèle`,
In: RAIRO, Vol 11, No 1, 1977.

[4] BODART, F., PIGNEUR, Y.
´A model and a language for functional specifications and evolution of information system dynamics`,
In: Formal Models and Practical Tools for Information System Design, H.S.Schneider (ed), North Holland Publishing Company, 1979.

[5] BODART, F.
´Logical specifications for system dynamics. An extension to PSL`,
ISDOS working paper No 183, April 1979.

[6] PETERSON, J.L.
´PETRI Nets`,
In: Computing Survey, Vol 9, No 3, 1977.

CHAPTER 3. SOME CONCEPTS AND PRINCIPLES FOR IMPLEMENTATION.

3.1. PRINCIPLES FOR THE CONTENTS AND SCOPE OF A CONCEPTUAL SCHEMA.

To design a conceptual schema, that is, to establish and describe the necessary propositions of the universe of discourse, the information system designer starts by observing the selected portion of the world constituting the universe of discourse and constructing in his mind an abstraction of it, much in the same way as a scientist, by experiment and analysis, constructs a theory of the observed phenomena.

As in science, we would normally expect that such an abstraction would be general enough to encompass many different (preferably all) occurrences of the same kind of phenomena. This abstraction constitutes propositions that necessarily hold in all possible entity worlds, just as the theory must hold for the observed or expected phenomena.

The process of observation, abstraction, and conceptual schema formulation is usually iterative. This is similar to the scientist´s attitude during the formulation of a theory: further experimentation - that is, observation of the universe of discourse - is generally required to clarify some aspects of the theory.

Although selection of what is considered to be necessary propositions about the universe of discourse to be described in the conceptual schema is to a certain extent arbitrary, as already is mentioned in chapter 1, the systems designer might consider various factors in deciding the boundary of the the conceptual schema.

An example of a widely accepted and also basic distinction has already been mentioned in chapter 1. It is the distinction between sentences which represent general laws and rules to which possible entities in the universe of discourse have to adhere, and sentences which describe facts about particular entities in the universe of discourse, following the laws and rules described in the conceptual schema. The main reasons for this distinction are practical arguments with respect to the design, implementation and maintenance of the information system, because some sentences are generalizations (conceptual schema) and others are specifics (information base). These generalizations have already been listed in chapter 1, section 1.8, as:

- describing classes (types, variables) in the universe of discourse rather than individuals (instances),
- describing concepts that are less subject to change rather than concepts that are changing more frequently,
- inclusion of rules or constraints having wide influence on

the behaviour of the universe of discourse (and therefore on the behaviour of the conceptual schema and information base) rather than narrow influence.

The guiding principle in selecting the necessary propositions is that they, although stating something about the universe of discourse, are convenient in controlling the consistency of the collection of sentences in the information base and the permissibility of their manipulation. As already stated in section 1.8 of chapter 1, the following general principles for the conceptual schema should be observed at all times:

* 100 Percent principle:
  All relevant general static and dynamic aspects, i.e. all rules, laws, etc., of the universe of discourse should be described in the conceptual schema. The information system cannot be held responsible for not meeting those described elsewhere, including in particular those in application programs.

* Conceptualization principle:
  A conceptual schema should only include conceptually relevant aspects, both static and dynamic, of the universe of discourse, thus excluding all aspects of (external or internal) data representation, physical data organization and access as well as all aspects of particular external user representation such as message formats, data structures, etc.

Some justifications and remarks may amplify the two principles:

Most of the general rules of the universe of discourse are currently described in application programs, if they are described at all. Sometimes such rules are referred to by the term "validation rules" in the current EDP jargon. In many situations, more than one update program operates on the same data base. In this case a single rule has to be described in each application program which may affect a concrete physical representation of that part of the information base covered by the rule. This results in a redundant representation of the rule, which is source for inconsistency among the various "copies" of the rule. It is obvious that due to such redundancy it is difficult to control, verify, and maintain a set of interrelated rules. It is even more difficult to detect contradictions in such a set of dispersed rules.

User update and query languages are becoming available which provide the (end) user tools to express his requests directly to the machine as opposed to a professional application programmer who writes the program on behalf of the employing user. One does not need too much imagination to think of situations where a user "has no time" to program the rules in his direct instruction of the machine. This may result in a stream of messages entering the information base that should not enter. The result is information base pollution. If all rules are expressed in the conceptual schema, and therefore are controllable by system functions independent of users or application programs, then there is no basis for such pollution.

If all general rules of the universe of discourse are expressed in the conceptual schema, then it is (much) easier to extend or modify these rules in a controlled way as compared to the situation where the rules are scattered over

several application programs. The need for such extension and modification is a normal thing to expect in practice.

It is easier to understand and to teach information systems design where all rules are in the conceptual schema, because of the absence of the previously mentioned kind of redundancy.

A conclusion of the 100 Percent Principle is that a conceptual schema language must be able to describe any set of general rules, etc. of the universe of discourse in a conceptual schema. In order to achieve this, some of these rules may be described in a procedural way while others may be described in a declarative way. This depends on the present state of the art of formal languages.

The Conceptualization Principle says that the conceptual schema must only and exclusively include conceptually relevant aspects of the universe of discourse. Aspects, constructs, or distinctions, which refer to other components or factors of the information system are not allowed to be part of the conceptual schema. Such aspects are, for example:

- Representation aspects of data in the user views;
- Aspects of machine efficiency and physical data organization;
- Organizational aspects of the information system.

By focussing on the conceptually relevant aspects, the conceptual schema design process is relieved of the burden of computer implementation aspects. The design process is in this way significantly simplified. More precise and accurate conceptual schemata will be the result of the conceptual schema design process. Moreover the design processes of the (external) application programs and the (internal) data storage routines will also be simplified as these can be limited to the external and internal representation aspects respectively.

Furthermore, by avoiding aspects of machine efficiency and physical data organization in the conceptual schema, operations at the user interfaces can be made completely independent of those "internal" aspects. Thus, reprogramming of applications in case of physical data reorganizations can be avoided, due to the "isolating" effect of the conceptual schema.

Finally, the observation of this principle also results in better evolution, simply because there is no burden of conceptually irrelevant aspects which could increase the reprogramming effort in cases of conceptual schema changes.

3.2. PRINCIPLES FOR THE DESCRIPTION OF A UNIVERSE OF DISCOURSE.

The theoretical foundation for describing a universe of discourse is, as already mentioned chapter 1, section 1.7, the use of an interpreted formal system of logic. The establishment of such a formal system of logic to describe the universe of discourse requires:

1. The determination of an alphabet for which it is unambiguously and mechanically recognizable whether a given character is in the alphabet, conditions naturally satisfied by conventional character input to a digital data processing system;

2. The provision of a finite set of effective rules (algorithms) that determine which strings of characters are to be taken as well-formed and, in particular, which well-formed strings are to be taken as sentences;

3. The provision of a finite set of effective rules that determine whether a given sentence is an axiom;

4. The provision of a finite set of effective rules that determine whether a given sentence can be immediately deduced from a given set of sentences;

5. The provision of a finite set of rules of interpretation that assign to each sentence a meaning such that the sentence is interpreted as an assertion unambiguously true or false about the universe of discourse.

In this context an axiom is defined as:

AXIOM
Any closed sentence that is asserted to be considered as such by an authorized source.

The axioms and deduction rules are chosen in such a way as to result in:

- each axiom being interpreted as a true assertion about the universe of discourse,

- each sentence immediately deducible from a set of sentences interpreted as true assertions about the universe of discourse being itself interpreted as a true assertion about the universe of discourse.

There is, of course, nothing unusual in these principles. They are simply a precise explication of the ordinary process of deductive inference couched in terms of a formal system so that they may be applied in the context of digital information systems.

Points 1 - 4 above relate to the syntax or grammar of a conceptual schema and information base. Principles of an abstract syntax are discussed in section 3.3, so that any formal language adhering to this abstract syntax can be used to carry out the purposes described in this chapter. Point 5 above relates to the semantics of the conceptual schema and information base. The principles for the semantics are discussed in section 3.4 with a similar objective.

A suitable formal system of logic to describe a universe of discourse may place minimal demands, in principle, on the part of the conceptual schema built in the information processor [1]. That is, the built-in minimal conceptual schema can be limited to bare essentials. In practice, however, many linguistic constructs, that in theory might be unnecessary, may very well be included in an actual information processor for reasons of efficiency, convenience and costs. Moreover, it is recognized that human perceptions of the universe of discourse, as well as communication of those perceptions to others, may differ considerably from individual to individual.

The information system designer must be free to express the conceptual schema

in terms appropriate to the specific universe of discourse of concern and to the user's perception of it. Specifically, no constraints are to be imposed on the entities assumed to exist in the universe of discourse or on the properties they may be asserted to possess. The information system designer is constrained only by the requirement to adhere to the basic principles of logic built into the information processor and to the fundamental need for maintaining consistency.

Considering also that conceptual schemata must be easy to use and to understand by a variety of users, a mechanism for adding linguistic constructs of arbitrary complexity ("macro constructs") to the formal system is necessary so that users of the system may interact with it at any desired level of aggregation of constructs [2, 3, 4]. In order for this principle to be observed without violating the first principle, the formal system must have within itself a mechanism for incorporating the definitions of new constructs in terms of those already present.

Taking into account the 100 Percent and Conceptualization principles, we can already deduce some general guidelines for a system of linguistic constructs for describing universes of discourse. Even then we have the choice among a variety of linguistic constructs. Each of those, however, has ultimately to be based on the fundamentals of formal logic.

In order to illustrate the choices, we list a few examples:

One choice is, to set up a system of concepts without defining dedicated constructs, i.e. elementary constructs that express the fundamentals of formal logic are applied only for the description of a universe of discourse. The major advantage of such a "fundamentalistic approach" is that one can rely upon the long tradition of formal logic. However, it is questionable whether this choice does not overstrain the capabilities of the average information system designer. To overcome this problem the introduction of suitable macro-constructs may be useful.

For example, a well-known possibility might be to distinguish propositions stating the type or class membership of entities and to design constructs to easily express such propositions. Many persons who are involved in practical conceptual schema design appreciate the merits of having a distinct notion of entity-type.

In many existing systems of constructs for conceptual schemata, there is also the distinction of sentences describing all classes of additional propositions possible for all entity worlds of the universe of discourse, and other sentences describing all relevant restrictions with respect to the allowed populations of the former type of propositions. Again, the reason to make this distinction has to do with practical considerations: In conceptual schema design, the determination of types (classes) usually and naturally comes before the determination of restrictions of their populations.

Furthermore, the required language must reflect the concepts established in section 2.2 of chapter 2: command-statement and action-description for insertion, deletion, retrieval, modification, etc. and suitable constructs for expressing the dynamic rules and command-conditions, as outlined in chapter 2, section 2.7 and 2.8.

3.3. ABSTRACT SYNTAX FOR A CONCEPTUAL SCHEMA AND INFORMATION BASE.

Precise definition of a universe of discourse, or, indeed, any precise dialogue about a universe of discourse, whether to a human or a computer, requires a language. In the present context, since communication with a computerised information system is essential, the language is necessarily formal. For a language to be formal it is not enough that its grammar and meaning be precisely specified. It is also essential that there be a decision procedure (an algorithm) which can, by examination of the individual alphabetic characters in an expression, determine whether the expression is grammatically correct in the language, and furthermore what kind of language constructs are involved in the expression. For example, it must be possible to determine which expressions are names, which are sentences, what sequences establish deducibility of a sentence, etc. In such a case a computer process can parse the language completely.

There are many ways of defining the syntax of a formal language, such as Backus-Naur form, production rules, the method chosen in appendix C, and so on. In order not to prejudge the grammatical form of any conceptual schema language, this Report specifies the syntax by defining the grammatical notions without specifying any rules about how these notions might be expressed in any particular language. These notions are: terms and sentences as defined in chapter 2, section 2.1, and grammatical functions called functors [5]:

> FUNCTOR
> A linguistic object that refers to a function on other linguistic objects taking as arguments (input) a list of linguistic objects (terms, sentences, functors) and yielding as a value (output) a single, uniquely determined linguistic object (term, sentence, functor).

Terms and sentences may involve variables:

> VARIABLE
> A term which refers to unspecified, indeterminate entities in the universe of discourse.

Sentences may be either open or closed. For example, the sentence

"x is a manufacturer."

which cannot be said to be true or false unless it is known to what x refers. Such sentences are said to be open sentences.

The sentences

"Ford is a manufacturer."

and

"for all x, if x is a manufacturer, then x is a company."

are, however, either true or false, depending on the universe of discourse. Such sentences are called closed sentences. Only closed sentences are unambiguously true or false and for that reason will be the only kind of sentences appearing in a conceptual schema and information base.

3.4. SEMANTICS OF A CONCEPTUAL SCHEMA AND INFORMATION BASE.

To attribute meaning (semantics) to various expressions in a language as mentioned above, it is necessary to start with a (hopefully small) set of undefined concepts known as primitives. Other concepts then have meanings which are derivable from the informally understood primitive concepts through the introduction of formal definitions. The relevant essentials of the meaning of each primitive concept is formally captured through the assertion of axioms assumed to be true.

The informal definition of truth we have adopted is:

> A sentence asserts a true proposition if it asserts that the state of affairs (of the universe of discourse) is so-and-so, and the state of affairs (of the universe of discourse) is so-and-so.

It is well beyond the scope of this chapter to elaborate this informal characterization - it took Tarski [6] some 80 pages - but its essence should be clear.

The interpretation of sentences depends on the meaning of the terms and predicates constituting the sentences. The interpretation of terms follow from the definition: they refer to particular entities in the universe of discourse. The interpretation of variables, the only primitive terms, has to be understood as "any entity".

The key to the semantics is the interpretation of the predicates in the sentences. To understand (be able to interpret) a predicate in a sentence is to be able to determine for any list of entities (keeping in mind that entities are part of the universe of discourse, not the terms in the sentences referring to them) whether or not the proposition expressed by the sentence is true of that list of entities.

In principle, of course, this is quite straightforward; it is not always so in practice. One of the design tricks in the development of a conceptual schema and information base will be to choose predicates asserting propositions where this task of interpretation is relatively easy.

Having chosen predicates appropriate to the universe of discourse, thus permitting the assertion of anything one wishes to say about it, there remains the task of choosing the axioms and the rules of deduction. As noted above, the axioms must all assert true propositions and the deduction rules must preserve truth.

For most real situations this is not as difficult as it might seem. Most of the problems relating to the generation of contradiction occur when infinite collections are contemplated and the axioms dealing with them, as such, are part of mathematical logic. While it is known that one cannot prove consistency (Gödel [7]), long use of standard axioms suggests that in practical cases there will be no problem.

3.5. PRINCIPLES FOR THE COMPOSITION OF CONCEPTUAL SCHEMATA.

Three main issues arise in discussing principles for the composition of a conceptual schema and information base:

1. What goes into the conceptual schema;
2. What are the abilities of the information processor;
3. How does the information processor react with the conceptual schema.

These three issues are evidently closely interconnected, as discussed at some length in sections 2.3 and 2.4 of chapter 2.

The information processor has a built-in minimal conceptual schema that cannot be changed without changing the information processor itself. Any suitable collection of sentences consistent with this minimal conceptual schema may be inserted using the mechanisms of the information processor in order to extend this minimal conceptual schema to form an actual conceptual schema. This process can be applied recursively.

Therefore, the conceptual schema for an information system in practice can be perceived as being built up like some sort of onion, the centre of the onion being formed by the minimal conceptual schema, the extensions representing the coats of the onion. The centre and inner coats symbolize the conceptual schema for the outer coats.

In the sense of this onion view - see figure 3.1 - any conceptual schema can be built up in a systematic fashion using the capabilities of the information processor as they are defined in the minimal conceptual schema. In the same fashion the information processor will "know" at each instant in time which part of the actual collection of sentences, constituting the conceptual schema and information base, it must consult as "conceptual schema" to control the manipulation of other sentences. Important decisions in the modelling and design process are what sentences will be in the minimal conceptual schema, what sentences will be in inner coats, etc.

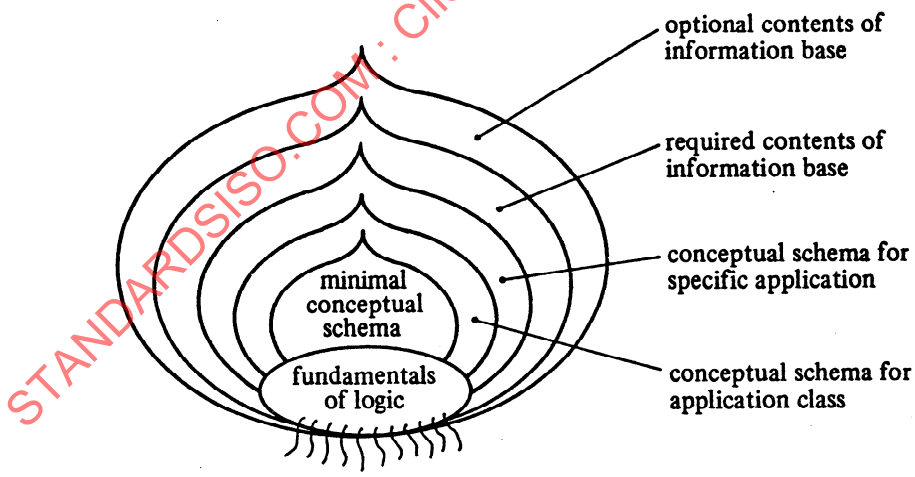


Figure 3.1. The onion-view for conceptual schema composition.

In section 3.2 it is already mentioned that this minimal conceptual schema really may be an absolute bare minimum. In practice, however, many more axioms and derived constructs will be built-in. These certainly will include the basic

axioms of mathematics and logic as provided by the hardware and basic software of the computers to be used. They also will include the axioms and derived constructs as expressed in the language constructs of the chosen basic language for the conceptual schema and information base. This collection of axioms and derived constructs expresses a collection of necessary proposition basic for almost any universe of discourse.

For many similar applications universes of discourse will certainly be relevant for which the same necessary propositions will hold. Therefore, the collection of necessary propositions common for such applications, and not already expressed in the minimal conceptual schema, might be expressed in axioms and derived sentences in an inner "coat" of the conceptual schema. Examples are common axioms for e.g. banking applications, personnel applications, goods movement applications, etc. It is imaginable that such conceptual schema extensions might be provided by your friendly software vendor.

The axioms and derived sentences expressing the necessary propositions specific for a particular universe of discourse might form the outer "coats" of a conceptual schema. As such they will be formulated in the design process of the conceptual schema for a particular information system. Probably also, these will be most likely subject to change in cases where a conceptual schema has to be changed.

As already discussed in chapter 2, section 2.1, the information base is the one consistent collection of sentences, that is, consistent with itself and the conceptual schema, that expresses the additional propositions of interest for the relevant entity world described in the information system.

Within the information base the concept of onion coats can be continued, as e.g. in the case that some set of sentences is required to be in the information base. This is a different concept from the axioms or the sentences deducible therefrom that express necessary propositions. For example, a necessary proposition about our example universe of discourse (see appendix B) requires that for each year of interest a maximum rate of fuel consumption is established. This will be expressed by a sentence in the conceptual schema. Sentences expressing the rates for specific years may be in the information base, provided the year is involved in the entity world of interest. Such sentences may be considered to be in an inner coat of the information base.

Given the conceptual schema and possibly a collection of essential or required sentences, the rest of the information base is optional, providing only that the totality forms a consistent collection. At any given time, of course, the instantaneous state of the conceptual schema and information base is a precisely defined collection of sentences.

3.6. THE THREE LEVEL ARCHITECTURE.

As a direct consequence of the conceptualization principle it is necessary to provide any implementation of an information system with at least two types of interface. The first type of interface, towards the users of the information system, deals with the external representation forms convenient in a particular user view.

These external interfaces are described in external schemata:

EXTERNAL SCHEMA
The definition of the external representation forms for the possible collections of sentences within the scope of a particular user's view including the manipulation aspects of these forms.

An external interface is the actual interface between a user in the environment and the implemented information system.

The second type of interface, to the storage facilities of the computer on which the information system is implemented, deals with such matters as:

* the internal (physical) representation forms of the information within the computer and on the storage media, etc.;
* computer process efficiency and efficient access mechanisms to the stored data;
* control of concurrent use, recovery, etc.

These interfaces are defined in internal schemata:

INTERNAL SCHEMA
The definition of the internal representation forms within the computer for the possible collections of sentences that are in the conceptual schema and information base including the manipulation aspects of these forms.

An internal interface is the interface between the implemented information system and the actual physical data storage facilities.

A general architecture, as outlined in the ANSI/SPARC Reports [8, 9], may be perceived as follows:

Figure 3.2. Inserting new information (conceptual view).

As described in section 2.3 of chapter 2, a message is presented to the information system containing e.g. new information to be inserted in the information base. The information processor, controlled by the rules as described in the conceptual schema and possibly by other sentences already present in the information base, will either insert the information or reject the message generating an appropriate message to report the result. Other actions will be dealt with in a similar way. This conceptual view on the information system has been depicted in figure 3.2.

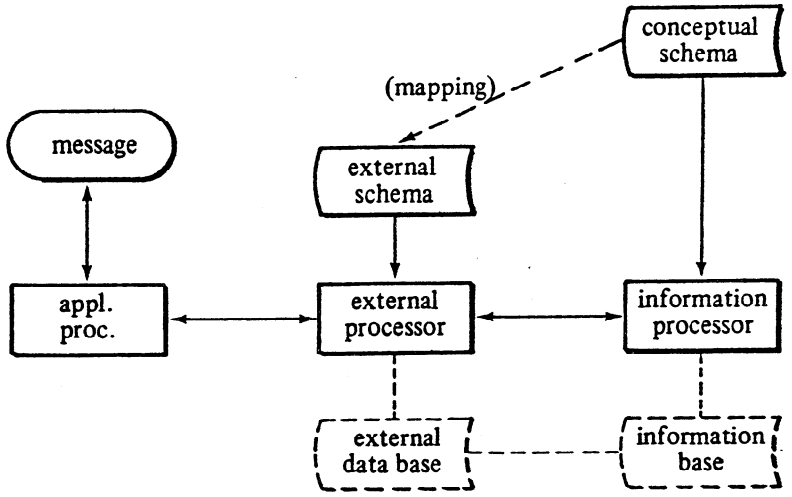


Figure 3.3. Inserting new information (external + conceptual view).

In practice the user is only interested in his external view of the information. That is, he is only interested in a subset of the information base in a representation form that he has defined as convenient for him. This implies that his application process dealing with the message has a particular external view of data (strings of characters) constituting his external data base representing the relevant information. This particular external view is described in the external schema relevant for the application process. However such an external data base is a virtual one mapped into the (relevant part of the) information base. This implies that the information system must handle:

* the "integration" of the actions of the various users;

* the transformation of their particular external views to the common (conceptual) view known to the information system.

It remains the task of the information processor to enforce the rules defined in the conceptual schema and to take care of the insertion. The additional tasks are performed by the external processor. Therefore the external schemata also contain the transformation rules. This has been depicted in figure 3.3.

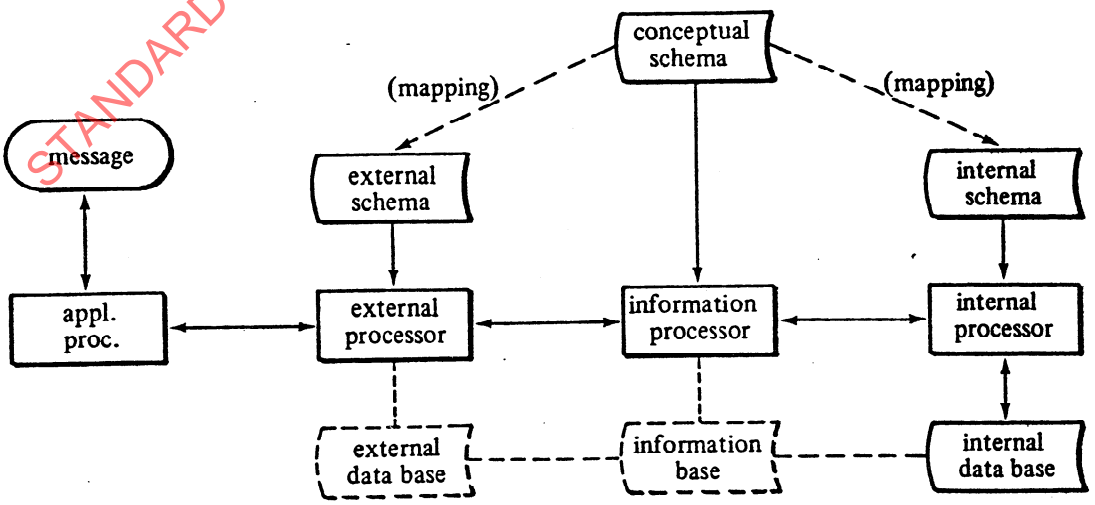


Figure 3.4. Inserting new information (external, conceptual, internal view).

The information base is also in practice a virtual one. The information is actually represented within the computer system by internally stored physical data forms (records, segments, fields, etc.) in the internal data base. These forms are declared in the internal schema. This implies an additional transformation process that is performed by an internal processor. The transformation rules for this are also described in the internal schema. This has been depicted in figure 3.4.

The internal data base itself is dealt with by the data storage facilities of the computer system.

According to the three schemata principle both the external and internal schemata and processors allow multiple layers. Also the internal data base may be implemented as a family of internal data bases, each of them "storing" a part of the information base. Such data bases may overlap (cf. distributed data bases).

As a slight variant to this ANSI/SPARC approach one may, at the conceptual level, describe a particular user's view - his particular universe of discourse - in a conceptual subschema (cf. the cover plate "The Metaphor of the Searchlights"). Several user views may be unified in a conceptual subschema with wider scope. For example, the views of several departments can be combined in an integrated view at group level within the enterprise. The conceptual schema in the information system describes the "union" of these various conceptual subschemata. At the conceptual level representation forms are not relevant.

At the external level, representation forms that are convenient for the user are defined and described in external schemata. Each user view (conceptual subschema) is mapped into one or more external schemata defining the appropriate representation forms and each describing an external data base that is assumed to exist within the scope of that user view, although in a virtual form.

In the case where an external view is a union of several views (e.g. the external data bases of individual departments are clustered into one common data base at group level), the resulting external schema will encompass several individual external schemata and describe a common data base in an "external but unified" form. The subsetting function of the external schemata is maintained and controlled by the external processor.

The external data bases are mapped into physical data bases. Although often this mapping may be to one physical data base, this need not necessarily be so. Several external data bases may be mapped into one physical data base; one external data base may be mapped into several physical data bases, or any other combination. Distributed networks may be involved.

A physical data base is defined in an internal schema. The transformation from external to internal form is in principle done by the internal processor. In the case of distributed data bases, the interconnection between external and internal data bases may be described in a distribution schema which might be part of the (unified) external schema(ta) interfacing with the internal schemata.

The rule enforcement task of the information processor may be implemented in a set of procedures. These need not necessarily be executed by one dedicated (information) processor in between the external processor(s) and internal processor(s). Especially in case of distributed data base systems the procedures may be distributed over the relevant external and internal processors involved.

This results in a three level architecture, which may be perceived as given in figure 3.5:

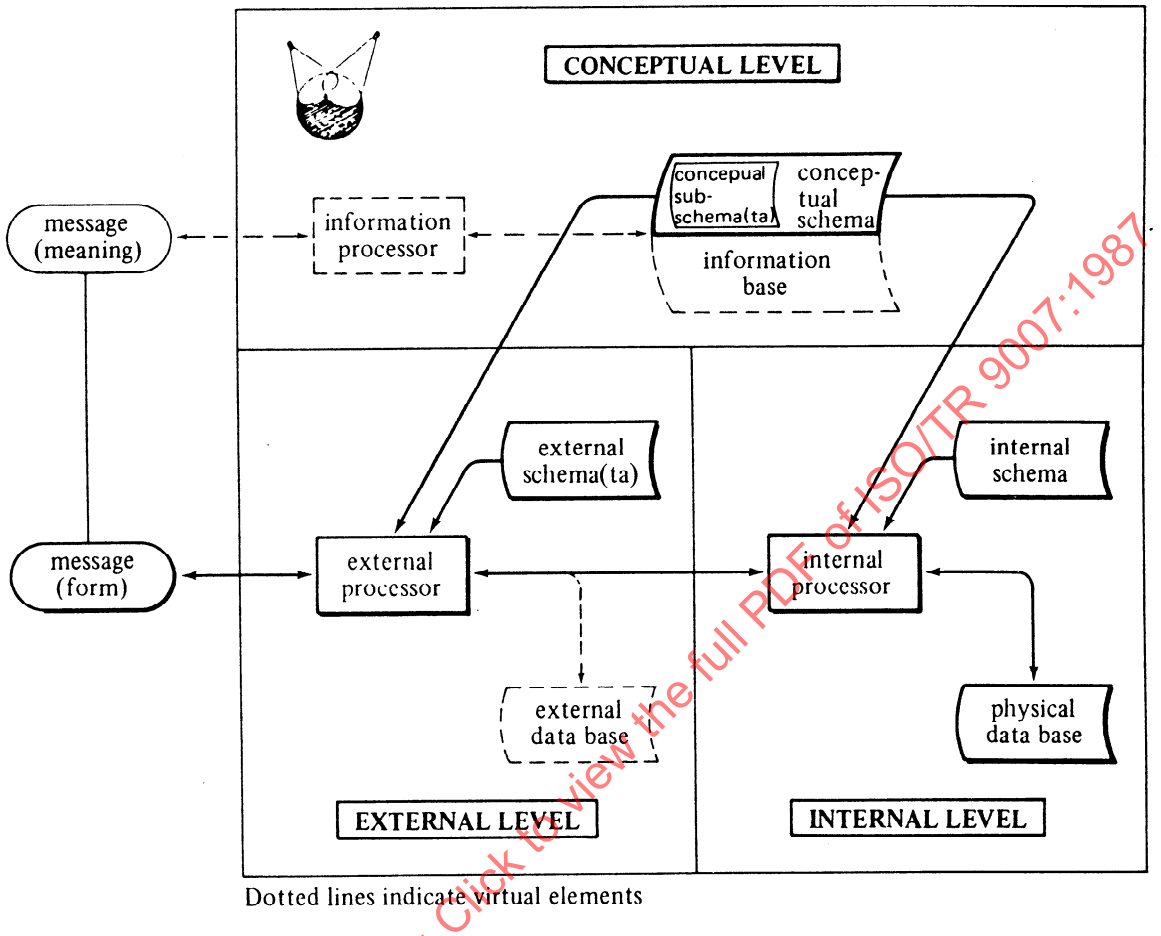


Figure 3.5. Three Level Architecture of an information system.

The main issue is that the conceptual schema is considered at all times to be a description of the necessary propositions for the universe of discourse and therefore controls what is described in the information base, not how it is described. The conceptual schema controls the semantic meaning of all representations, that is, defines the set of checking, generating, and deducing procedures of the information at the conceptual level in the information system. However, it does not describe an intermediate state in the transformation process from external to internal forms.

The external schemata describe how the users wish to have the information represented. The external processor interfaces directly with the users and coordinates their information exchanges.

The internal schemata describe the internal physical representation of the information. The transformation between external forms and internal forms is done, in principle, by the internal processor(s). Therefore the external processor(s) interface with the internal processor(s). The mapping between the external schemata and the internal schemata must preserve meaning as defined by the conceptual schema.

3.7. INFORMATION RESOURCE DICTIONARY SYSTEM (IRDS) MODEL.

Information about the information system itself is most important for the users of such a system, for the designers, and for those responsible for its proper functioning. Hence there is great interest in data dictionary systems.

Such a dictionary system for an information system is actually an information system in itself. It describes a universe of discourse consisting of the first mentioned target information system, including all implementation aspects and (parts of) its environment.

The ANSI X3H4 (IRDS Technical Committee) is in the process of defining a standard proposal for such systems, which they call: "Information Resource Dictionary Systems (IRDS)". We will gladly adopt this term.

The conceptual IRDS architecture can be perceived as in figure 3.6.

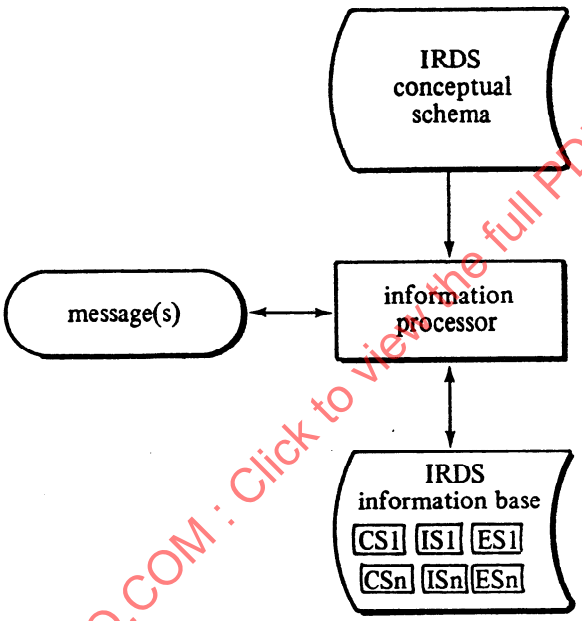


Figure 3.6. Information Resource Dictionary System Architecture.

In this architecture the conceptual schema for an IRDS describes the universe of discourse containing all possible target information systems that may be documented using the IRDS. Accordingly, the IRDS conceptual schema provides a description of what can be described in the IRDS information base and the permissible actions on the IRDS information base.

The IRDS information base describes all target information systems for which the IRDS is used, including the conceptual, external, and internal schemata, the various processors, programs, users, etc.

The above indicates that an IRDS and the target information system(s) it describes need not be disjoint - at least the schemata in the IRDS information base already overlap with the target information system(s). It is also quite possible that the same processor mechanisms may be used.

Points of further investigation in the very near future are in our opinion:

* The applicability of the conceptual schema and information base concepts to IRDS models;

* The requirement for future information system implementations to encompass an integrated IRDS.

3.8. THE CONCEPTUAL SCHEMA IN THE CONTEXT OF CURRENT DBMS IMPLEMENTATIONS.

As already mentioned the conceptual schema may be considered to have two principal purposes:

* describing the universe of discourse ("enterprise model");

* controlling the descriptions in the information base ("information/data base model").

The first purpose implies that the ways in which the conceptual schema is formulated are, in principle, independent of computer implementations. The second purpose, according to some authors, requires the conceptual schema formulation be directed towards the computer oriented data structures and constructs. Other authors and current implementations [e.g. 10, 11, 12, 13] apply a fourth kind of schema in order to "interface" with constructs of a current DBMS:

The conceptual schema is dedicated to the first purpose mentioned and hence defines the semantics of the information base and, therefore, the interpretation of all representation forms thereof. This conceptual view is (manually) converted to a "common data base view" expressed in convenient computer oriented data structures and constructs. The model of this data base is formulated in a fourth schema, the data base schema, that fulfils the second purpose mentioned in this section. It is supported by a data base processor.

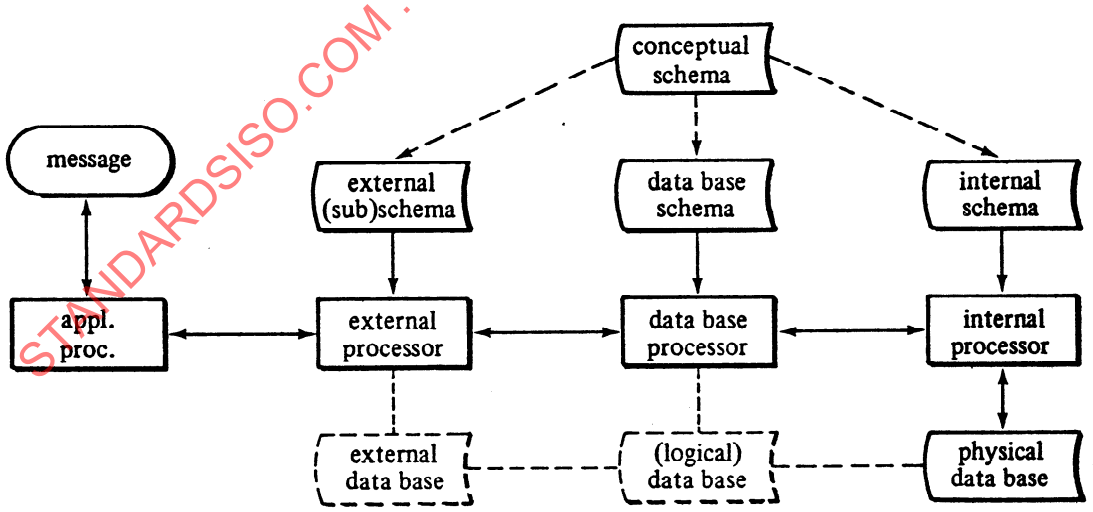


Figure 3.7. Inserting new information (applying a fourth schema concept).

Such a data base schema may be needed if the DBMS used does not implement all features needed at the conceptual level or if they are implemented in computer

technology oriented constructs. This architecture is depicted in figure 3.7. Some examples of a translation into constructs of the current CODASYL proposals and into some relational implementations may be found in [11].

In most cases the conceptual schema is partly covered by the (logical) data base schema and a set of rule enforcing procedures (data base procedures) that are called by the data base processor. In these cases this data base schema has a more or less "internal" oriented character.

3.9. CORRESPONDENCE OF THE THREE LEVEL ARCHITECTURE FOR INFORMATION SYSTEMS AND THE REFERENCE MODEL FOR OPEN SYSTEMS INTERCONNECTION.

The Reference Model for Open Systems Interconnection (OSI), advocated by ISO TC97/SC16, describes how communication occurs between information systems (application processes) using the open systems interconnection mechanism [14]. The model is divided into seven functional partitions, called layers, whose names are the application, presentation, session, transport, network, data link, and physical layers.

The purpose of the application layer is to serve as the window between communicating users of the OSI environment through which all exchange of information meaningful to the users occur.

The purpose of the presentation layer is to represent information [in external representation forms] to communicating users in a way that preserves meaning while resolving syntax differences.

The purpose of the session layer is to provide the means necessary for co-operating units of the presentation layer to organize and synchronize their dialogue and manage their [external] data exchange. To do this the session layer provides services to establish session-connection between two presentation layer units, and to support their orderly data exchange interaction.

The transport layer, and the underlying network, data link and physical layers provide the computer, data network, and data communication technical facilities to be used by the presentation layer.

The Three Level Architecture for information systems, discussed in section 3.6, relates to the OSI Reference Model in the following way (see figure 3.8):

The conceptual and external levels of the Three Level Architecture for information systems are related to the functions of the application and presentation layers respectively. It is possible that some of the concepts developed for conceptual schemata and information bases could be applied to specification of the semantics of information traffic in those layers. However, the exact nature of the relationship of the conceptual and external levels to the specific functions of the application and presentation layers remains to be specified.

The internal level deals with the internal storage representation, including the internal data manipulation and actual physical storage on the storage media. In this its realm of operations is comparable to the character of the realm of the session, transport, network, data and physical layers. Their functions, however, being storage and communication respectively, are different.

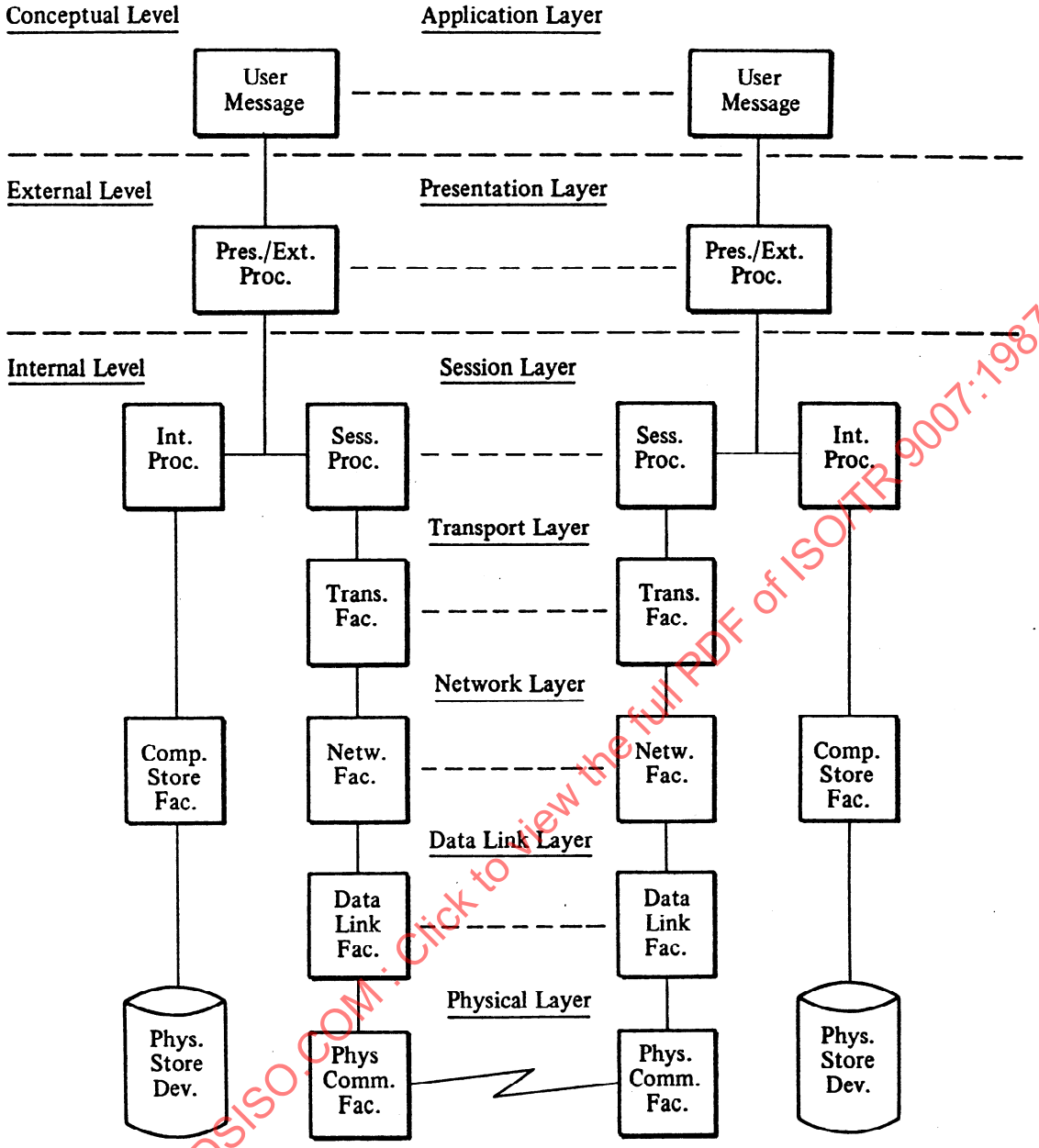


Figure 3.8. Layer architecture of OSI compared with three level architecture.

Points of (joint) investigation might be in our opinion:

* The degree of coincidence, correspondence, or applicability of concepts of OSI in the information system as outlined in this Report and vice versa;

* The applicability of (selected) OSI concepts for information systems applying internally to communication facilities (distributed information systems).

3.10. REFERENCES.

[1] STEEL, T.B., Jr. 'A Modest Proposal for a Conceptual Schema Language', In: Proceedings of SHARE LI, vol. 1, SHARE Inc, Chicago, 1978.

[2] FALKENBERG, E., BREUTMAN, B. and MAUER, R.
´CSL: a language for defining conceptual schemas`,
In: Data Base Architecture, Proceedings IFIP TC2, Venice 1979, North-Holland Publishing Company.

[3] NIJSSEN, G.M.
´Information Analysis`
Research Report, CONTROL DATA, 1979.

[4] van GRIETHUYSEN, J.J.
´Notation Methods for an Information Model`
Internal Report, N.V. PHILIPS, 1978, 1981.

[5] ANDERSON, A.R., BELNAP, N.D.,
´Entailment - The logic of relevance and necessity.`,
Princeton University Press, 1975.

[6] TARSKI, A.
´Logic, Semantics, Metamathematics`,
Oxford University Press, 1956 (originally in Polish and German, 1933-1936).

[7] GÖDEL, K.
´Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I`,
In: Monatshefte für Mathematik und Physik, 37, pp. 173-198, 1931.

[8] ANSI/X3/SPARC,
´Study Group on Data Base Management Systems: Interim Report 75-02-08`,
In: ACM SIGMOD Newsletter, FDT, Vol 7, No. 2, 1975.

[9] TSICHRITZIS, D., and KLUG, A. (eds.)
´The ANSI/X3/SPARC DBMS Framework. Report of Study Group on Data Base Management Systems`,
AFIPS Press, Montvale NJ, 1977.

[10] KENT, W.
´Splitting the conceptual schema`,
In: Proc. Sixth Intl. Conf. on Very Large Data Bases, Montreal, Oct. 1-3, 1980.

[11] OLLE, T.W.
´Multistage Data Definition in a multicomponent DBMS architecture`,
In: B.Schneiderman, Data Bases: Improving usability and responsiveness, Academic Press, 1978.

[12] LEVIN, M., SCHNEIDER, L.S.
´The separation of enterprise and data models`,
Data base research project, Sterling Systems Inc., Golden Colorado.

[13] van GRIETHUYSEN, J.J., van MELIS, W.H.P.M., van RIEL, A.J.J.,
´Data organization in DB/DC-systems`
Internal Report, N.V.PHILIPS, 1981.

[14] ISO TC97/SC16
´Data Processing - Open System Interconnection - Basic Reference Model`,
Document DP 7498, ISO TC97/SC16, 1981.

CHAPTER 4. OVERVIEW OF SOME MODELLING APPROACHES.

4.1. INTRODUCTION.

Research work in the past decade in the field of modelling for data bases has produced a number of papers, each expounding the merits of a specific method for conceptual schema models. In an attempt to bring a better understanding of modelling approaches, we have tried to identify some general aspects in the various methods, and to roughly characterize them in this chapter. Our investigation is not exhaustive. The sole purpose is to provide some convenient criteria with the objective of:

- o identifying various techniques for describing a universe of discourse in a conceptual schema and information base;
- o identifying fundamental concepts necessary for conceptual schema languages;
- o establishing criteria for analysing and judging present and future candidates for conceptual schema language standardization.

For the purpose of this chapter as outlined above we have selected the following criteria:

1. Form versus meaning:
Earlier methods concentrated on the forms of the data that was modelled. That is, structural forms for data were defined, which would be convenient for storage and/or manipulation in a computer. In particular, the update possibilities were optimized, although the access path structures, important for retrieval, were emphasized as well. The term data modelling stems from these methods.

More recent modelling techniques stress the importance of modelling the meaning (semantics) of the information. The semantic rules for pieces of information play important roles in these methods. Such semantic models are considered to be independent of, but paramount to, the data models that describe the representation and storage forms of the information. In other words, these semantic models support the conceptual view as identified in the ANSI/SPARC reports. The term information modelling is often associated with these techniques.

Proponents of these semantic model methods in no way neglect the data forms and their influence on practical performance problems. Their contention, however, is that one can deal properly with the data storage and manipulation, and the performance requirements in particular, only if there is a clear understanding - and formal definition - of exactly what the data represents, which semantic rules and constraints exist, and what information manipulation actions are needed by the user.

2. Static aspects versus dynamic aspects:
Many of the modelling methods concentrate on the static aspects of the conceptual schema and information base. Others emphasize especially the dynamic aspects or sometimes are almost fully operation oriented. Some models include all aspects.

3. Capability of distinguishing between lexical and non-lexical entities:
Many methods do not allow a clear distinction between lexical and non-lexical entities, i.e. the names of things and the things themselves. Others permit the distinction or even require explicit distinction between them.

4. Expressive power:
By this we mean the degree of completeness with which a given method can formally express all applicable aspects and constraints of the universe of discourse in the conceptual schema. The expressive power may differ significantly among the methods.

5. Information granularity:
Some of the methods deal with constructs which refer to single, semantically independent propositions of the universe of discourse - thus stating every proposition explicitly as a separate granule. On the other hand, some methods allow also for constructs expressing propositions of arbitrary complexity that group several simple propositions in one granule.

6. Distinction of various kinds of propositions:
Some of the modelling methods handle all propositions in the same manner, while others distinguish various kinds of propositions, thus handling (and designating) them differently. For example, some modelling methods distinguish between attributes of entities and relationships among entities, while others consider this distinction to be irrelevant at the conceptual level. As another example, some methods associate a distinct kind of propositions with the notion of types, considering them as basic propositions to be treated in a specific way.

4.2. REVIEW OF SOME APPROACHES.

Several modelling methods for information systems and data bases exist today. A partial list includes, but is not limited to the following methods (listed alphabetically):

* Abstract data types,
* Binary relationships models,
* Conceptual graphs,
* Deep structure sentence models,
* Entity - relationship models,
* Function-oriented or operations-oriented models
* N-ary relationships models,
* Network models (including CODASYL)

* Object - role models,

* Process-interaction models,

* Relational models

* Semantic nets

* Set theoretic models,

We should note that each of these methods has a number of advocates and each is a distinct way of viewing the problem of conceptual schemata and information bases. It remains as a research question whether these methods are different in essential ways, or are in some real sense equivalent. An answer to that question is not essential for the present purpose.

However, as a first attempt, we have tried to identify groups of more or less similar methods based on basic concepts and characteristics. The following three groups of approaches have been selected for discussion and illustration in this Report:

- Entity-Attribute-Relationship approaches, (section 4.2.1)

- Binary and Elementary N-ary Relationship approaches, (section 4.2.2)

- Interpreted Predicate Logic approaches. (section 4.2.3)

We have not tried to classify any of the listed methods or any others into the above mentioned groups of approaches. It is quite possible that some methods can be considered as belonging to more than one group of approaches. It is also possible that other groups of approaches may be perceived. It remains as future work to determine whether any of these above mentioned approaches are stand alone approaches, or contain any of the others, and whether still other groups should be discerned.

We also wish to emphasize that none of these approaches is sufficiently described in enough detail to be considered as a candidate for standardization, although this observation should not be taken as a prejudgement on the possibility that one or more of the approaches might lead to the presentation of a candidate for a conceptual schema language.

A single example universe of discourse is used for demonstration purposes in each of the approaches described in the appendices. This example universe of discourse together with a possible entity world is described in appendix B.

In the discussions of the various approaches some sort of formal languages are used to demonstrate the essential aspects. Although even the "grammars" of these demonstration languages are given, this should not be construed as a proposal for such a language to be candidate for a conceptual schema language. They also do not suggest any form that the Working Group feels is suitable or convenient for such languages.

The "syntaxes" of these demonstration languages have been uniformely described in a specific syntax notation. This must not be considered as a suggestion for this syntax notation to be superior to any other. For the reader's convenience, however, a short summary of this syntax notation is given in appendix C.

4.2.1. ENTITY ATTRIBUTE RELATIONSHIP APPROACHES.

The EAR approaches were introduced to the data base community and further developped by authors such as Bachman and Chen, and are based on the use of the following concepts:

- entities,
- relationships among entities.
- attributes, associations between values and entities, or between values and relationships.
- values,

These approaches also make use of the notions of type and occurrence applied to each of its primitive concepts.

The origins of these approaches are the data modelling practices of the early seventies. Originally only binary (dyadic) relationships were allowed and attributes of relationships were not recognized. However, more recent developments have resulted in variants that allow n-ary relationships between entities and allow relationships to have attributes.

The EAR approaches can be characterized as being oriented towards the definition of static aspects. Therefore, generally speaking they can describe only partially the various rules of the universe of discourse. The EAR approaches often imply special kinds of propositions that are grouped together and expressed in single macro constructs. They do not provide for explicit distinction between lexical and non-lexical entities.

An outline of the EAR approaches is presented in appendix D.

4.2.2. THE BINARY AND ELEMENTARY N-ARY RELATIONSHIP APPROACHES.

Historically, the binary relationship approaches have their roots in certain approaches in artificial intelligence and linguisitics, dealing with "semantic networks" and other similar notions. They were introduced to the data base community in the early seventies by authors such as Abrial, and further developed by several other authors in the mid-seventies (e.g. Senko, Bracchi).

The BR approaches distinguish entities from entity-names. They do not distinguish between attributes and relationships. Furthermore, only relationships that are binary are recognized. The BR approaches are based on three primitive concepts:

- entities,
- entity-names,
- binary relationships.

Also these approaches make use of the notions of type and occurrence applied to each of its primitive concepts.

The Binary Relationship approaches started out with the capability of defining mainly static aspects, but in recent years they have been extended to handle dynamic aspects as well. These approaches are now able to describe all rules that are relevant for the universe of discourse. Variants of these approaches distinguish explicitly between lexical and non-lexical entities.

Developments with the same roots and philosophy, also in the mid seventies, resulted in the elementary n-ary relationship approaches (e.g. Falkenberg). These approaches do not restrict an elementary proposition to be about exactly two entities, but allow description of elementary propositions involving one, two or more entities (elementary n-ary relationships).

The basic idea of all these binary and n-ary relationship approaches is to model the universe of discourse explicitly and separately using sentences that express simple elementary propositions, thus not introducing a specific grouping of those elements. Grouping is not considered to be at the conceptual level of the conceptual schema.

For the purpose of illustration we restrict ourselves in this Report to Binary Relationship approaches, thus following the work of Abrial, Bracchi, and Senko. However, much of what is discussed is equally well applicable to elementary n-ary relationship approaches.

An outline of the BR appraoches is presented in appendix E.

4.2.3. INTERPRETED PREDICATE LOGIC APPROACHES.

The IPL approaches, as proposed by authors such as Steel, perceive the universe of discourse as solely consisting of entities, for which propositions hold. The conceptual schema and information base constitute a description consisting solely of a set of sentences encoded in some formal language based on formal logic. Such sentences are composed of:

- terms and variables
- predicates
- logical connectives
- quantifiers

The terms and variables refer to the entities in the universe of discourse and the sentences express the propositions about those entities.

The essence of the approaches is the establishment of an interpreted, axiomatized, deductive, formal system of logic describing the universe of discourse without placing any modelling constraint on the universe of discourse itself.

The basic principles of these approaches are equally well applied to both static and dynamic aspects of the universe of discourse and those of its description in the conceptual schema and information base. Therefore the approaches are able to describe all the rules as prescribed for the universe of discourse and ipso facto its description. They also provide for explicit distinctions between lexical and non-lexical entities.

Some variants of the approaches apply a very limited set of elementary constructs that are built into the information processor, and use these to "generate" and construct the full conceptual schema and information base relevant for the chosen universe of discourse. Others include more complex constructs and capabilities in their basic set to increase the ease and convenience of a user in expressing all kinds of propositions about the universe of discourse. They all provide for dynamic change of the conceptual schema as well as the information base. Several of them also provide for dynamically extending the expressive power of the language used by adding possible constructs using the capabilities that are already present.

An outline of the IPL approaches is presented in more detail in Appendix F.

4.3. TRANSLATION OF APPROACHES TO CURRENT DATA BASE TECHNOLOGY.

We believe that it is possible to find translation rules for translating approaches into other approaches, although in some cases such translations may be only partial. However, we consider it even more important for obvious and practical reasons to translate (implement) conceptual schema, information base, and information system requirements in today´s existing data base technology.

For example, practitioners of data base technology are interested in exploring how the conceptual schema framework fits into their own data base world, i.e. how the conceptual schema can be transformed in their existing data base management facilities like CODASYL Systems, Relational Systems, File Systems, Hierarchical Systems, etc. (cf. also chapter 3, section 3.8).

This transformation from conceptual schema to existing Data Base Management Systems depends on the ability or facilities that are available in those DBMSs to enforce the rules and constraints as declared in the conceptual schema, i.e. the "schema" facilities. Current practice and most available DBMS software limits us to enforcing a major portion of the conceptual schema rules via application programs (cf. chapters 1 and 3 on the 100% principle).

Taking into account the software technology already known today, we foresee future work on software systems that will be able to almost completely enforce the conceptual schema rules automatically - thus avoiding the enforcement of these conceptual schema rules by the application programs.

Given, however, the situation today where the application program has the responsibility of enforcing a major part of the conceptual schema rules and constraints, if enforced at all, the data base designer should first properly and formally define the conceptual schema (in either one of the described or other suitable approaches). This would enable him to first define what the problem is by means of the descriptions in the conceptual schema, then to define how the problem is to be implemented by which software. By doing this he provides for the proper definition of the problem, and he makes it relatively easier to foresee possible problem areas or changes in the future and even to plan for the changes.

Some parts - usually small parts - of a particular conceptual schema, defined according to some suitable approach, can be mapped directly into the data structures according to the "schema" facilities of various of today´s DBMSs, at least in principle (see figure 4.1). Even within one given DBMS "schema" facility, there may be various ways of performing this mapping, thus the data

base designer has the possibility of selecting - starting from one conceptual schema - that data base schema which best fits other requirements, e.g. efficiency of data manipulation.

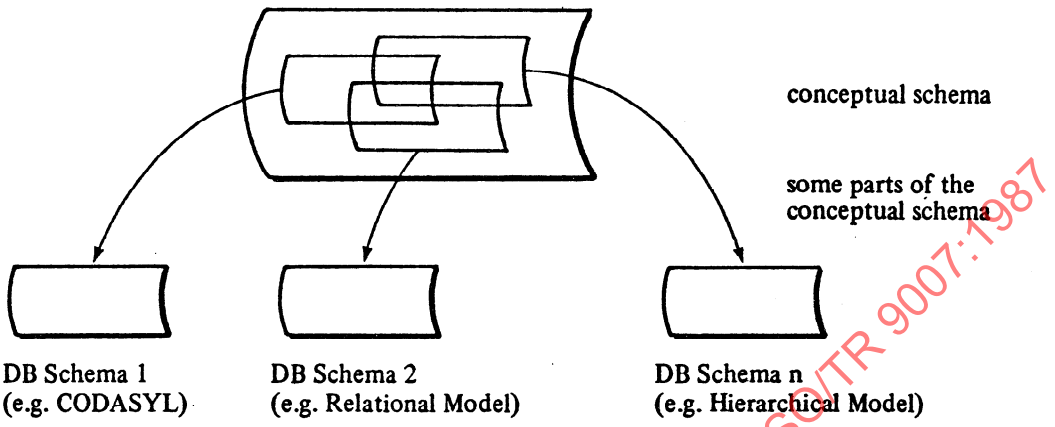


Figure 4.1. Translating a conceptual schema into a data base schema.

Other parts of the conceptual schema, especially many of the (more complex) static and dynamic constraints, cannot be mapped (expressed) in today's data base schema languages. Therefore they can only be enforced by defining procedures that are called either by the DBMS (data base procedures) or by application programs.

The ease of translating or mapping the conceptual schema to a data base implementation, which may be called schema mapping flexibility, may differ considerably among different approaches and various DBMSs.

In order to show, how this mapping can be performed in principle, we give the following examples of mapping aspects.

1. Mapping of conceptual schema constructs.

Starting with a conceptual schema, where the elementary granules are already grouped in a certain way, as in the case with an EAR approach, a fairly obvious and straight-forward kind of mapping is always possible. For example, the collection of single-valued attributes of an entity becomes a record in a conventional data base approach; a functional relationship between two entities becomes a "link" between the corresponding two records, while a non-functional relationship becomes a record.

2. Grouping of elementary propositions.

Beginning with a conceptual schema where only elementary granules of information are defined, as in the binary and elementary n-ary relationship approaches and several variants of the IPL approaches, a large variety of different possiblities of grouping those elements into larger constructs usually exists.

For example, various kinds of record structures or relational data structures can be defined upon that basis, and so achieve a high degree of schema

mapping flexibility. The data base designer can select that particular structure which fits within the given schema facility, and which takes into account e.g. efficiency considerations.

Various rules and algorithms are known which assist in this transformation enabling the data base designer e.g. to easily isolate the applicable "keys" or "identifiers", "candidate keys" in the "record types" and "n-ary relations" as well as to obtain a normal form of update anomaly-free data structure.

3. Mapping of static constraints.

Some of these can be implemented by using capabilities such as validation definitions of the target "schema" system. For example, some uniqueness constraints may be directly transformed as "keys" in the target "schema" system. For enforcing more complex constraints application dependent checking procedures might be linked to the DBMS system software, if the DBMS provides for such possibilities (e.g. data base procedures for which the call is defined in the data base schema).

4. Mapping of dynamic constraints.

This can be implemented partially by current "schema" systems. For example, the CODASYL DDL AUTOMATIC membership clause, "subset", "structural constraint", or "application-written-facilities-to-perform-the-automatic-function" in other systems, can be used to enforce some of the dynamic rules and constraints of the conceptual schema. Many of the others can be implemented by data base procedure options as indicated above.

4.4. REFERENCES.

Appropriate references are listed at the end of appendices D, E, and F.

APPENDIX A. GLOSSARY OF TERMINOLOGY AND DEFINITIONS.

ACTION
One or more elementary actions that, as a unit, change a collection of sentences into another collection of sentences in the information base or conceptual schema and/or make known a collection of sentences present in the information base or conceptual schema.

ACTION DESCRIPTION
A linguistic object describing an action or permissible action.

ACTUAL ENTITY WORLD
A collection of entities of interest that is described in an actual information base and its conceptual schema.

ACTUAL INFORMATION BASE
That information base which exists in a specified instant or a period of time, usually referred to as "now", and which expresses the additional propositions other than the necessary ones, that hold for an entity world.

AXIOM
Any closed sentence that is asserted to be considered as such by an authorized source.

CLASS (of entities)
All possible entities in the universe of discourse for which a given proposition holds.

COMMAND
The order or trigger for an action or permissible action to take place.

COMMAND CONDITION
The precondition, including synchronization aspects, that must be met before a permissible action may take place.

COMMAND STATEMENT
A linguistic object expressing a command or elementary command.

CONCEPTUAL LEVEL
All aspects which deal with the interpretation (meaning) and manipulation of information describing a universe of discourse or entity world in an information system.

CONCEPTUAL SCHEMA
A consistent collection of sentences expressing the necessary propositions that hold for a universe of discourse.

CONCEPTUAL SCHEMA LANGUAGE
A formal language, parsable by a computer as well as a human being, containing all linguistic constructs necessary to formulate the sentences in a conceptual schema and an information base and their manipulation in terms of action-descriptions, command-conditions, etc.

CONCEPTUAL SUBSCHEMA
A consistent collection of sentences expressing the necessary propositions that hold for a universe of discourse that is limited to a particular user's view and as such is part of a conceptual schema relevant for the (shared) information system.

CONCEPTUALIZATION PRINCIPLE
A conceptual schema should only include conceptually relevant aspects, both static and dynamic, of the universe of discourse, thus excluding all aspects of (external or internal) data representation, physical data organization and access as well as all aspects of particular external user representation such as message formats, data structures, etc.

DATA
The representation forms of information dealt with by information systems and users thereof.

DATA BASE
The representation of all information that is dealt with in an information system, taken together.

DATA BASE SCHEMA
The definition of the representation forms and structure of a data base for the possible collections of all sentences that are in the conceptual schema and information base including manipulation aspects of these forms.

DATA BASE SYSTEM
The computer implementation of an information system.

DELETION
The removal of a previously inserted sentence from the information base or conceptual schema. Any deducible sentence, which cannot be deduced without the deleted sentence, will no longer be deducible and therefore no longer be a deducible part of the information base or conceptual schema.

ELEMENTARY ACTION
The insertion, deletion, or retrieval of a sentence.

ELEMENTARY COMMAND
The order or trigger for an elementary action to take place.

ENTITY
Any concrete or abstract thing of interest, including associations among things.

ENTITY WORLD
A possible collection of entities that are perceived together.

EVENT
The fact that something has happened in either the universe of discourse, or the environment, or in the information system.

ENVIRONMENT
That part of the real world containing the users which exchange messages with the information system.

EXTERNAL EVENT
An event that occurs in the environment or universe of discourse.

EXTERNAL LEVEL
All aspects dealing with the user-oriented representation of information visible at the outer interfaces of an information system.

EXTERNAL SCHEMA
The definition of the external representation forms for the possible collections of sentences within the scope of a particular user's view including the manipulation aspects of these forms.

FUNCTOR
A linguistic object that refers to a function on other linguistic objects taking as arguments (input) a list of linguistic objects (terms, sentences, functors) and yielding as a value (output) a single, uniquely determined linguistic object (term, sentence, functor).

HELSINKI PRINCIPLE
These utterances are to be interpreted (recursively) as international English utterances:
Any meaningful exchange of utterances depends upon the prior existence of an agreed set of semantic and syntactic rules. The recipients of the utterances must use only these rules to interpret the received utterances, if it is to mean the same as that which was meant by the utterer.

HOMONYMS
Identical terms that refer to different entities.

INFORMATION
Any kind of knowledge about things, facts, concepts, etc. of a universe of discourse that is exchangable among users. Although exchangable information necessarily will have a representation form to make it communicable, it is the interpretation of this representation (the meaning) that is relevant in the first place.

INFORMATION BASE
A collection of sentences, consistent with each other and with the conceptual schema, expressing the propositions other than the necessary propositions that hold for a specific entity world.

INFORMATION PROCESSOR
The mechanism that in response to a command executes an action on the conceptual schema and/or information base.

INFORMATION RESOURCE DICTIONARY SYSTEM
An information system dealing with the information about a universe of discourse consisting of another (target) information system, its environment as far as relevant, and its implementation in a data base system. It is not necessarily disjoint from the target information system.

INFORMATION SYSTEM
The conceptual schema, information base and information processor, forming together a formal, fully predictable system for keeping and manipulating information.

INSERTION
The addition of a sentence to the information base or conceptual schema. Other sentences, not deducible before insertion may become deducible and therefore become a deducible part of the information base or conceptual schema.

INSTANCE (of an entity-type)
An individual entity, for which a particular type proposition holds, that is, which belongs to a particular class of entities.

INTERNAL EVENT
An event that occurs because of the termination of some permissible action in the information sysem.

INTERNAL LEVEL
All aspects dealing with the user-transparent representation of information within the computer physical implementation of an information system.

INTERNAL SCHEMA
The definition of the internal representation forms within the computer for the possible collections of sentences that are in the conceptual schema and information base including the manipulation aspects of these forms.

LEXICAL OBJECT
A (simple) linguistic object that is used only to refer to an entity.

LINGUISTIC OBJECT
A grammatically allowable construct in a language.

MESSAGE
A collection of one or more sentences and/or command statements to be used as an information exchange between the environment and the information system.

MODIFICATION
The replacement of a sentence in the information base or conceptual schema by another one, thereby possibly changing the collection of sentences which are deducible.

NAME
A (simple) linguistic object that is used only to refer to an entity.

NECESSARY PROPOSITION
A proposition asserted to hold for all entity worlds and therefore must be part of all possible proposition worlds.

OCCURRENCE (of an entity-type)
An individual entity, for which a particular type proposition holds, that is, which belongs to a particular class of entities.

100 PERCENT PRINCIPLE
All relevant general static and dynamic aspects, i.e. all rules, laws, etc., of the universe of discourse should be described in the conceptual schema. The information system cannot be held responsible for not meeting those described elsewhere, including in particular those in application programs.

PERMISSIBLE ACTION
An action, conforming to specified rules or constraints, which

- changes a presumably consistent collection of sentences in the information base or conceptual schema into a consistent collection of sentences

and/or

- makes known a consistent collection of sentences present in the information base or conceptual schema.

PREDICATE
A linguistic object, analogous to a verb, which says something about an entity or entities to which term(s) in the sentence refer.

PROPOSITION
A conceivable state of affairs concerning entities about which it is possible to assert or deny that such a state of affairs holds for these entities.

PROPOSITION WORLD
A collection of propositions each of which holds for a given entity world.

RETRIEVAL
To make known a sentence which has been inserted in the information base or conceptual schema, or is deducible from other sentences in the information base or conceptual schema.

SENTENCE
A linguistic object which expresses a proposition.

SYNONYMS
Different terms that refer to the same entity.

TERM
A linguistic object that refers to an entity.

TYPE (of an entity)
The proposition establishing that an entity is a member of a particular class of entities, implying as well that there is such a class of entities.

UNIVERSE OF DISCOURSE
All those entities of interest that have been, are, or ever might be.

USER
Anybody or anything that issues commands and messages to the information system and receives messages from the information system.

VARIABLE
A term which refers to unspecified, indeterminate entities in the universe of discourse.

EXAMPLE UNIVERSE OF DISCOURSE.

B.1. INTRODUCTION.

The example which follows has been created expressly for the purpose of having a single universe of discourse of at least moderate plausibility which can be described by each of the various approaches. It is sufficiently small so that a large description is not required, but sufficiently complex to exhibit the essential differences among the various approaches. In addition, there is enough dynamic specification to permit the same universe of discourse to be used to exemplify the dynamic aspects.

In section B.2 we present a prose description of the classifications and rules for our example universe of discourse in natural English language. This could be considered as an informal conceptual schema.

We also give - in section B.3 - some example entities and happenings in our universe of discourse. This could be regarded as an informal (and incomplete) information base describing an entity world of interest.

B.2. RULES, ETC. FOR THE UNIVERSE OF DISCOURSE.

The universe of discourse to be described has to do with the registration of cars and is limited to the scope of interest of the Registration Authority. The Registration Authority exists for the purpose of:

* Knowing who is or was the registered owner of a car at any time from construction to destruction of the car.

* To monitor certain laws, for example regarding fuel consumption of cars and their transfer of ownership.

Manufacturers of cars:

There are a number of manufacturers, each with one unique name. Manufacturers may start operation, with the permission of the Registration Authority (which permission cannot be withdrawn). No more than five manufacturers may be in operation at any time. A manufacturer may cease to operate provided he owns no cars, in which case permission to operate lapses.

Cars:

A car is of a particular model and is given a serial number by its manufacturer that is unique among the cars made by that manufacturer. The manufacturer is registered as the owner of the car as soon as practicable. At this time it is given one registration number, unique for all cars and for all time. The year of production is also recorded. During the month of January only, a car may be declared to have been produced in the previous year. Eventually a car is destroyed and the date of destruction is registered. The history of a car must be kept until the end of the second calendar year following its destruction.

Car models:

A model of car has one universally unique name. Cars of each model are made by only one manufacturer. New models may be introduced without limit. All cars of one model are recorded as having the same fuel consumption.

Fuel consumption:

Fuel consumption is a number of litres of hydrocarbon fuel per 100 kilometres, which will be between 4 and 25 litres. The fuel consumption averaged over all registered cars produced by a particular manufacturer in a particular year is required not to exceed a maximum value, which is the same for each manufacturer and may change from year to year. At the end of each January an appropriate message is sent by the Registration Authority to each manufacturer which has failed to meet this requirement.

Garages:

There are a number of garages, each with one unique name. New garages may start trading. Garages may own cars, but at any time the cars they own must have originated from no more than three manufacturers (which three is unimportant, and may vary with time). A garage cannot cease to trade as long as it owns cars.

Persons:

There are a number of persons who can own one or more cars. Each person has one unique name. Only those persons are of interest who own, or have at some time owned, a car still known to the Registration Authority.

Car ownership:

At any time a car may be owned by either its manufacturer, or a trading garage, or a person or group of persons. If a car is owned by a group of persons, each is regarded as an owner.

Transfer of ownership:

Ownership of a car is transferred by registration of the actual transfer, including the date. A manufacturer can transfer only to garages, and cannot be a transferee. A garage can transfer only to people. After destruction of the car it cannot be transfered anymore. Earlier transfer though still can be recorded.

There are no specifications except as above.

In the example, it is inevitable that a number of simplifying assumptions have been made, e.g. reasons, prices and circumstances of transfer of ownership are not considered.

B.3. SOME THINGS AND HAPPENINGS IN THE RELEVANT ENTITY WORLD.

The following describes some of the things and happenings of the assumed entity world:

Some of the car manufacturers with permission to operate are Ford, General Motors, Renault, Jowett, and Volkswagen. A couple of Ford's models are the Mustang and the Granada. General Motors constructs, among others, the Impala model.

The history of a Ford Mustang, serial number PCXX999 is as follows. It was constructed in 1975 and presented for registration on 21 January 1975. It got the registration number GMF 117. On 29 January 1975 it was distributed to Smith's garage, which sold it on 15 March 1975 to Mr. Johnson. Mr. Baker bought the car from Mr. Johnson on 24 May 1978. The car was destroyed on 13 January 1980.

A General Motors Impala car, serial number QGTM783F, was registered under number ABC 653 on 9 April 1978 and distributed to Jones Brothers Ltd. This car was bought by Mr. Johnson on 26 May 1978. The car was destroyed on 14 August 1979.

General Motors made an Impala car with serial number QAVP864B in 1977. It was registered on 21 January 1978 and given registation number PQR 456. On 14 February 1978 it was sold to RN Cars who already owned other cars made by General Motors, Renault and Volkswagen. Mr. and Mrs. J. Soap bought this car on 31 March 1978, but were unable to trade in their Datsun as part of the deal.

In 1978, a new manufacturer PSC (Pretty Small Cars) requested permission to manufacture, and was refused. Following the failure of Jowett in 1979, the request was resubmitted, and this time PSC (Pretty Small Cars) got permission to operate on the market on 1 January 1980.

The first model produced was the Gasmiser. The first series of this model with serial numbers GAM1001, GAM1002, and GAM1003 were all registered on 4 January 1980. They got the registration numbers XYZ 101, XYZ 102, and XYZ 103 respectively.

The car XYZ 101 was distributed to the garage named South Station on 25 January 1980, but was destroyed by an accident on the same day. The latter two cars were distributed to the North Station Garage on 20 January 1980. Both cars were sold to Messrs. Gödel, Escher, and Bach on 26 January 1980. Mr. Bach died in an accident with car XYZ 103 on 2 March 1980. The car was registered as destroyed on 5 March 1980. On 5 March 1980 the Messrs. Gödel and Escher were registered as sole owners of the remaining car XYZ 102. They sold this car to Smith's garage on 15 march 1980 and bought from it a Mustang, serial number PCXX010, which was registered first on 5 January 1980, built in 1979. The registration number of this latter car was XYZ 109. XYZ 102 was destroyed by dismantling, because Smith's garage had a shortage of spare parts. Therefore at the end of 1982 the Registration Authority can forget all about the cars XYZ 101 - XYZ 103.

On 1 December 1980 PSC gave notice of withdrawal from the car business.

For 1979 the average fuel consumption was established as a maximum of 12 litres per 100 kilometres, for 1980 it was established as a maximum of 10 litres per 100 kilomitres, which will be the rate for 1981.

This page intentionally left blank

APPENDIX C. THE PASCAL SYNTAX NOTATION.

We have adopted the PASCAL syntax notation, as proposed by ISO TC97/SC5, for the languages used to demonstrate the example conceptual schemata in the various approaches. (See document ISO/TC97/SC5 N 678 Programming languages - PASCAL).

This syntax notation can be used quite well to describe itself, that is, to define the syntax notation. Note that the sequence in which the metastatements are written is immaterial. The only requirement for the description to be complete is that there must be a defining metastatement for every metavariable appearing in a metaexpression.

```
syntax              = metastatement {metastatement}.

metastatement       = gap metavariable gap "=" metaexpression gap ".".

metavariable        = letter {letter | digit | "-"}.

metaexpression      = sequence {or sequence}.

sequence            = gap element {comma element}.

element             = metavariable | iteration | option | nest |
                                                    terminal-symbol.

iteration           = "{" metaexpression gap "}".

option              = "[" metaexpression gap "]".

nest                = "(" metaexpression gap ")".

terminal-symbol     = string-description | terminal-character.

string-description  = string-delimiter string string-delimiter.

string-delimiter    = '"'.

string              = {terminal-character | any-character}.

terminal-character  = '''(any-character | '").

space               = " ".

gap                 = {space}.

or                  = gap "|".

comma               = gap [","  gap].
```

letter	denotes any letter of the alphabet, not further specified here.
digit	denotes any digit 0 through 9, not further specified here.
any-character	denotes any character except " in the character-set, not further defined here.

The semantics in general should be clear from the above description. The following remarks, however, might be useful:

metasymbol	meaning
=	shall be defined to be
|	alternatively
.	end of definition
option: [x]	0 or 1 instances of x
iteration: {x}	0 or more instances of x
nest: (x | y | z)	any one of x or y or z
terminal-symbol: "xyz"	the string xyz
terminal-character: ´x	the character x
´´	the character ´
string-delimiter: ´"	the character "

The quote (´) is to be interpreted as:

"Do not copy this character, but the next character as it stands."

The special terminal-character is needed to define:

- the character " in a string (´"),
- the character ´ in a string (´´),

because the empty string is defined as "".

Observe, "priorities of operation":

1. Any kind of brackets (option, iteration, nest).
2. Sequence.
3. Selection.

APPENDIX D. THE ENTITY - ATTRIBUTE - RELATIONSHIP APPROACHES.

D.1. EMPHASIS OF THE APPROACHES.

The Entity-Attribute-Relationship (EAR) approaches (sometimes also called Entity-Relationship approaches) have evolved from the work of Bachman [1] and Engles [2] on data modelling.

Since the appearance of Chen's paper [3], and the work of Tardieu and others [4, 5], there have been a number of papers exploring certain aspects of the approaches: for example, the possibility of modelling propositions about more than two entities by allowing n-ary relationship-types, and the possibility of relationships having attributes.

There exist many variants of the EAR approaches. While some of these variants are supported by available commercial software products, others form the basis for teaching courses (for example, [6, 7]). These variants have many aspects in common, for instance, the use of graphic formalisms to aid communications between systems designers and information user. Also, several alphabetical languages have been proposed, but there is no consensus on their syntax, and a considerable diversity exists in the detailed semantics.

The chief purpose of this chapter is to present the concepts, supported in most variants, in a way that explains the broad characteristics of these EAR approaches.

Considering the objectives for a conceptual schema, cited in section 1.9 of chapter 1, the following remarks could be made about these approaches:

1. "To provide a common basis for understanding the general behaviour of the universe of discourse."
   The use of the basic concepts - entity, attribute, and relationship - seems quite easy to understand and teach, although the right choice is often difficult to make when analysing the selected portion of the world constituting the universe of discourse. Emphasis is placed in particular on the relationship structure between entities.

2. "To define the allowed evolution and manipulation of the information about the universe of discourse."
   In the EAR approaches the use of the basic concepts provokes some rigidity as compared to other classes of approaches. For example, the distinction between attribute and relationship has the consequence that some difficulties may arise when changing the conceptual schema. As no dynamic constraints are formulated in most of the EAR approaches the control on allowed manipulation is rather limited.

3. "To provide a basis for interpretation of external and internal syntactical forms which represent the information about the universe of discourse."
   This role is not emphasized by most variants of the EAR approaches.

4. "To provide a basis of mappings between and among external and internal schemata."
In EAR approaches the role of a conceptual schema most frequently emphasized is to serve as a result of a stage in the interactive design process. In a later stage this conceptual schema is mapped manually into external and internal data models.

D.2. PRIMITIVE CONCEPTS OF THE APPROACHES.

D.2.1. THE BASIC CONCEPTS.

In the EAR approaches the universe of discourse is considered to consist of the basic concepts of entities which are said to have attributes, and relationships among entities. In some approaches the relationships as well are said to have attributes.

An entity, as defined in section 2.1 of chapter 2, is any concrete or abstract thing in the universe of discourse. For example, in a particular enterprise entities are a certain John Smith, a purchase order 75, and a Chevrolet auto with serial number 13750645W.

An attribute is a perceived property of an entity or a perceived property of an association among entities in the universe of discourse. The age of John Smith (34 years) may be considered an attribute of John Smith. Attributes are specific to the entity. For example, the age of John Smith and the age of the ship Cutty Sark are two different attributes.

An attribute is said to have a value. If in the above example the age of John Smith is an attribute, then 34 years is the value of that attribute.

A relationship is a perceived association between entities in the universe of discourse. For example, if John is an entity and Amsterdam is an entity, then the fact, that John lives in Amsterdam, is considered to be a relationship between the entity John and the entity Amsterdam.

In other words, both attributes and relationships are propositions about entities. Attributes in general are monadic or dyadic propositions about a single entity or relationship, often associating a particular value with that entity or relationship. Relationships in general are propositions about two or more entities (n-adic propositions). For example, John is married to Mary in Amsterdam. In some variants the relationship is restricted to propositions about two entities (dyadic or binary propositions).

D.2.2. ABSTRACTION CONCEPTS.

So far we have spoken primarily of instances of things. Things which have certain properties in common are said to be of a certain type. As already explained in section 2.1 of chapter 2 a type is a classification of similar things. The three basic abstractions of the EAR approaches are entity-type, attribute-type, and relationship-type. Some authors use the word "attribute-type" and "attribute" synonymously. We will use the word "attribute-type".

In the EAR approaches the notions of occurrence and population are frequently

used in correspondence with the type notion. An occurrence of a particular type has been defined already in section 2.1 of chapter 2 as a unique individual thing belonging to that type. The population of a type is a particular collection of occurrences of that type, and may vary from time to time. Usually it is the collection of all entities of the type that occur in the relevant entity world.

An entity-type is a classification of entities, each of which has similar attributes associated with it. Each occurrence of an entity-type must be unique and therefore distinguishable from all other occurrences of that entity-type. In the EAR approaches one distinguishes an entity by means of one or more attributes, called an identifier.

An attribute-type is a classification of similar attributes of all entity occurrences belonging to an entity-type or of all relationship-occurrences belonging to a relationship-type. An individual occurrence of an attribute-type, associated with the individual occurrence of an entity-type or relationship-type, is thought of as an (attribute) value. In this chapter from now on we will use the word "attribute-value". The <attribute-type, attribute-value> pair "identifies" the attribute for the entity. It is also said that the attribute-value(s) of an identifier identify the entity.

A relationship-type, in the EAR approaches, is a relationship defined over one or more entity-types. For example, an occurrence of the relationship-type LIVES-IN defined over the entity-types PERSON and TOWN may be expressed as "John lives in Amsterdam"; three occurrences of the relationship-type WORKS-FOR defined over the entity-types PERSON and DEPARTMENT may be expressed as "Tom works for Sales", "Dick works for Sales", and "Harry works for Sales". "John is married to Mary" expresses a relationship-occurrence of the relationship-type MARRIED-TO defined over the single entity-type PERSON involving two entity-occurrences. "John married Mary in Amsterdam" expresses a relationship-occurrence of the relationship-type MARRIED-IN defined over two entity-types PERSON and TOWN involving three entity-occurrences.

The concept of relationships among entities is fundamental to the EAR approaches. However it should not be assumed that an entity-type may participate in only one relationship-type. Any number of relationship-types may be defined in which the entity-type participates. Neither should it be assumed that only one relationship-type can be defined over a particular collection of entity-types. Any number of relationship-types may be defined involving the same collection of entity-types. In the case where two or more relationship-types are defined over the same collection of entity-types it is necessary to distinguish the relationship-types by assigning one or more names to it.

(Note: As will be explained in section D.3, we will adopt the convention to use capitals for terms referring to types such as entity-types and relationship-types.)

Some EAR approaches also identify value-types. A value-type is the classification of a collection of attribute-values, which may form pairs with a particular attribute-type so as to be attributes for entities of a specific entity-type or for relationships of a specific relationship-type. A value-type is always closely associated with one or more attribute-types. Another term for value-type, used by some authors, is domain.

In some EAR approaches entity-types are considered to be disjoint. That is, a

particular entity may be considered to belong to only one entity-type. In other cases entity-types need not be disjoint so that a particular entity may belong to more than one entity-type. The same observation applies to attribute-types, relationship-types, and value-types.

D.2.3. CHARACTERISTICS OF RELATIONSHIPS.

Differences exist among variants of EAR approaches especially with respect to characteristics of relationships. Therefore, we will discuss in the following each of the most important characteristics first for the general case, and thereafter point out what differences can occur in some variants.

In the EAR approaches relationships may be classified according to three criteria:

- o Dimension of the relationship-type,
- o The functionality of the relationship-type,
- o Total and partial relationships.

The dimension of a relationship-type is the number of entity-occurrences in a single occurrence of the relationship-type. The number of entity-occurrences is independent of the number of entity-types over which the relationship-type has been defined. For example, the relationship-types

WORKS-FOR defined over PERSON and DEPARTMENT
(e.g. "Tom works for Sales")

and

MANAGES defined over PERSON
(e.g. "Tom manages Harry")

both are relationship-types of dimension 2 (binary or dyadic relationship-type) defined over two entity-types in the first example, and defiend over a single entity-type in the latter case.

In general, a relationship-type of dimension n is called a n-ary relationship-type. Examples of a ternary relationship-type (dimension 3) and a quarternary relationship-type (dimension 4) are respectively

MARRIED-IN defined over two PERSONs and one TOWN:
e.g. "John married Mary in Amsterdam".

and

TRANSFER-MG defined over a MANUFACTURER, a GARAGE, a CAR, and a TRANSFER:
e.g. "Ford transfers to Smith's garage a car GMF 117 in the transfer of 29 January 1975".

Some EAR approaches are limited to binary relationship-types. Moreover, some of these only allow binary relationship-type over two different entity-types.

The functionality of a relationship-type is illustrated with respect to two distinct entity-types. The possible kinds of functionality for a binary relationship-type defined over entity-types A and B, are:

* one-to-one (1-1), in which one occurrence of A may bear only one relationship to one occurrence of B, and each occurrence of B may bear only one relationship to one occurrence of A;

* one-to-many (1-n), In which one occurrence of A may bear one or more relationships each to one occurrence of B, but each occurrence of B may bear only one relationship to one occurrence of A;

* many-to-many (m-n), in which one occurrence of A may bear one or more relationships each to one occurrence of B, and each occurrence of B may bear more than one relationship each to one occurrence of A.

The corresponding kinds of functionality also extend to n-ary relationship-types.

In practice this characteristic is considered together with the characteristic of total or partial relationship:

A total relationship defined over entity-types A and B requires that every occurrence of A and every occurrence of B must participate in a relationship of the relationship-type.

A partial relationship defined over entity-types A and B requires that some, but not all occurrences of A and some, but not all occurrences of B may participate in a relationship of the relationship-type.

The four possible cases which are entirely independent of the functionality of the relationship are:

A	B	
all	all	(total)
all	some	)
some	all	) (partial)
some	some	)

The combination of functionality and totality/partiality establishes the cardinality of the relationship, and is expressed in terms of min-cardinality and max-cardinality. A graphic notation for this is illustrated in figure D.1. A description of this graphic notation form is given in section D.4.

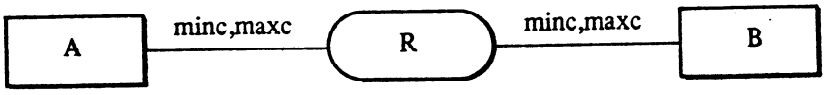


Figure D.1. General picture for a relationship-type.

The min-cardinality is the minimum number of times that each occurrence of an entity-type may be involved in a relationship of the relationship-type. The value 0 means that an occurrence may exist without being involved in any relationship of the relationship-type. The value 1 (or n) means that an entity-occurrence cannot exist without being involved in 1 (or n) relationship(s) of the relationship-type.

The max-cardinality is the maximum number of times that each occurrence of an entity-type can be involved in a relationship of the relationship-type. The value 1 means that an occurrence may be involved in at most one relationship of the relationship-type. The value n means that an entity-occurrence can be involved in n relationships of the relationship-type. Some examples are illustrated in figures D.2, D.3, D.4, and D.5.

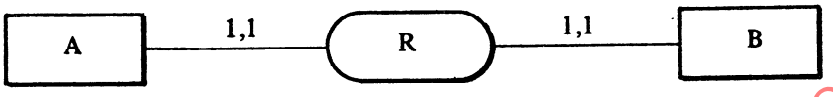


Figure D.2. A "one-to-one" relationship.

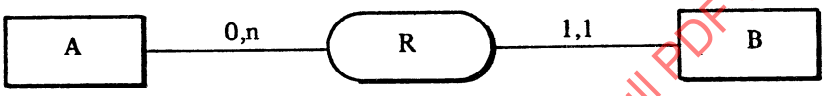


Figure D.3. A "one-to-many" relationship.

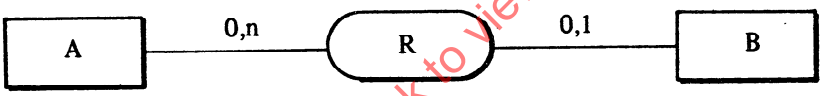


Figure D.4. A "zero or one-to-many" relationship.

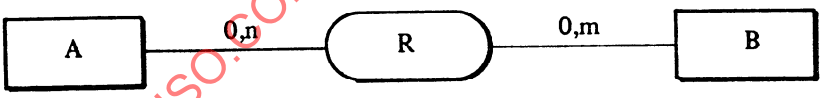


Figure D.5. A "many-to-many" relationship.

A classical example of the case "one-to-many" (figure D.6) is a relationship WORKS-FOR between a DEPARTMENT and a EMPLOYEE: A DEPARTMENT may have 0 to n EMPLOYEES working for it, a EMPLOYEE must be related to 1 and not more than 1 DEPARTMENT:

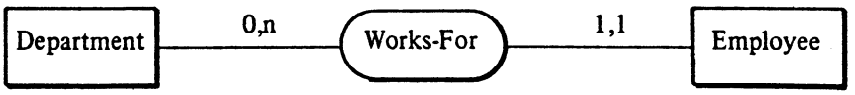


Figure D.6. A binary (n=2) relationship.

Another example may help to understand the cardinality on n-ary (n>2) relationship-types (figure D.7):

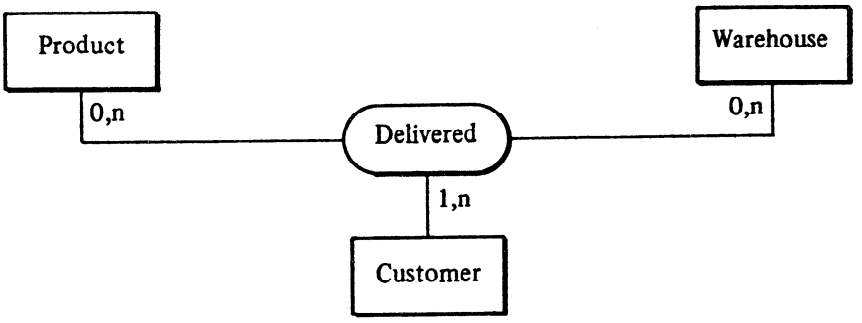


Figure D.7. A ternary (n=3) relationship.

A PRODUCT may or may not be involved in the relationship DELIVERED; if it is, it may be related to n DELIVERED relationships. A WAREHOUSE may or may not be involved in the relationship DELIVERED; if it is, it may be related to n DELIVERED relationships. An CUSTOMER is necessarily involved in at least one DELIVERED relationship, but may be involved in n DELIVERED relationships. A typical DELIVERED relationship relates one occurrence of CUSTOMER with one occurrence of PRODUCT and one occurrence of WAREHOUSE.

The functional dependency concept is applied only to the set of relationship-occurrences. It is possible though to define a functional dependency between a subset of the collection of entity-types and another entity-type.

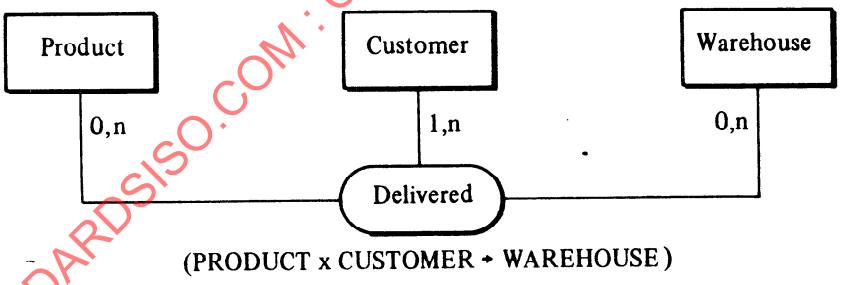


Figure D.8. Functional Dependency within a relationship-type.

The functional dependency (PRODUCT X CUSTOMER -> WAREHOUSE) in the example of figure D.8 expresses the constraint that for a given product and a given customer only one warehouse can deliver this product to this customer.

Some EAR approaches that only allow binary relationship-types restrict themselves to one-to-many (including one-to-one) and zero or one-to-many relationship-types between entity-types. Others also allow for many-to-many binary relationship-types.

In the EAR approaches, referred to in the previous paragraph, such one-to-many and zero or one-to-many relationship-types usually are depicted using diagrammatic techniques based on the Bachman diagrams. In these diagrams an entity-type is represented by a rectangular box, in wich the entity-type-name is written. A one-to-many relationship-type is represented by a complete arrow, the arrowhead pointing to the entity-type ("member" entity-type) that may play a part in "many" relationship-occurrences with a single entity ("owner" entity-type). A zero or one-to-many relationship type is represented by a dotted arrow. The relationship-type-names may be written next to the arrow, but they are often omitted. Some of the examples are given in figure D.9 as an illustration of the technique.

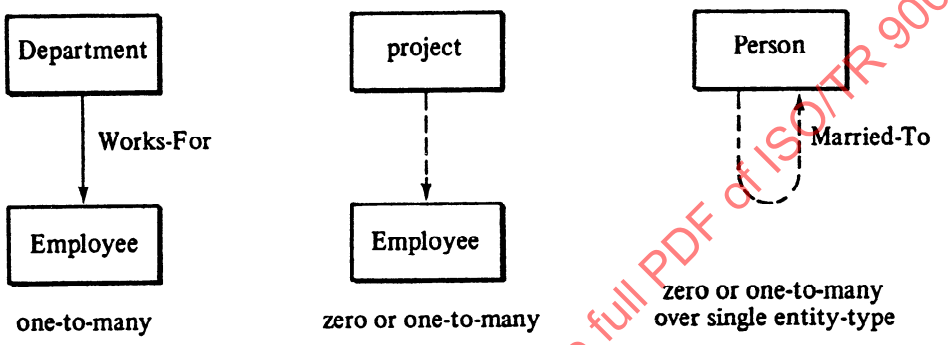


Figure D.9. Graphic representation of relationship types.

The example demonstrated in figure D.7 above may also be solved in the EAR approaches, that are restricted to binary relationship-types, by "objectifying" the relationship-type DELIVERED into the entity-type DELIVERY and using three binary relationship-types DELIVERED-BY, DELIVERED-FROM, and DELIVERED-TO, as shown in figure D.10:

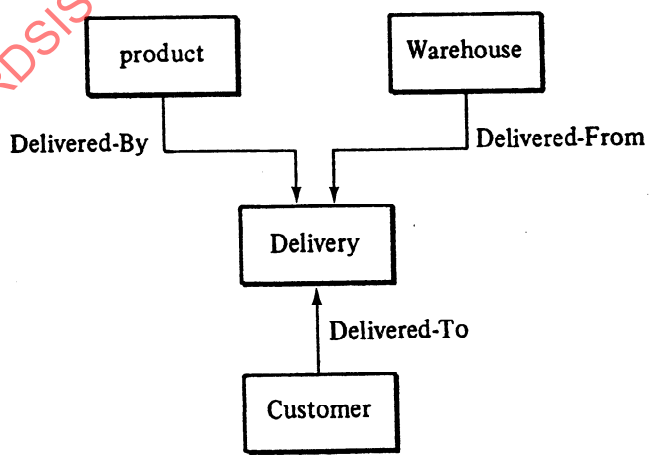


Figure D.10. Three binary relationships for the example of figure D.7.

In the general case not only entities, but relationships as well are said to have attributes. However, some EAR approaches, usually the ones that also restrict to binary relationships, do not allow attributes of relationships.

For the remainder of the discussion of the EAR approaches, and in particular for discussing the example in section D.6, we have chosen to demonstrate the general case, implying the following characteristics of the relationships:

* N-ary relationship-types are allowed;
* Many-to-many relationship-types are allowed;
* Attributes of relationships are allowed.

D.3. GRAMMAR AND SEMANTICS.

Until now we have considered the entities, attributes, etc., themselves in the universe of discourse. This, however, is a mental exercise. In chapter 2 we already pointed out that we need a language to communicate about a universe of discourse. That is, we need linguistic constructs to refer to and describe the entities, attributes, entity-types, etc. All EAR approaches have such constructs, sometimes explicitly defined, sometimes implicitly assumed. We intend to identify such constructs explicitly in this chapter.

The basic linguistic constructs are:

* Entity-type-names, which are lexical objects that refer to entity-types;

* Attribute-type-names, which are lexical objects that refer to attribute-types.

* Relationship-type-names, which are lexical objects that refer to relationship-types.

The approaches that also recognize value-types or domains additionally need:

* Value-type-names or domain-names, which are lexical objects that refer to value-types or domains.

We will adopt in this chapter the convention of using capitals for the above mentioned type-names.

In the literature, as already mentioned, many authors do not make clear distinction between "type" and "type-name". Some even go so far to use the word "entity-type", or still worse the word "entity", for the entity-type-name, causing much confusion. In this chapter though, PERSON is an entity-type-name - a lexical object - which refers to the class (entity-type) of all entities considered to be persons, etc.

In formal languages used in EAR approaches to describe a universe of discourse, more complex constructs are needed to describe the various types. We will use the general term description for these constructs: A description is a sort of graph, picture, or language construct which describes a type, that is, lists its characteristics. Note, that some authors use the word "type" for the notion of "type-description".

In the EAR approaches discussed in this chapter, we identify the following descriptions:

* entity-type-descriptions, listing the attribute-types and other characteristics, such as the identifier, for the entity-types;

* attribute-type-descriptions, listing the characteristics of attribute-types;

* relationship-type-descriptions, listing the entity-types over which the relationship-type has been defined, and additional characteristics, such as the category of the relationship-type, the cardinalities, and attribute-types, if any.

In the variants which recognize value-types or domains, these also have to be decribed using:

* value-type-descriptions or domain-descriptions, listing the characteristics of the value-types.

Examples will be discussed in section D.5 and following.

The syntax of the language used in this chapter to describe the conceptual schema for the example universe of discourse of appendix B is given below. However, as already stated, this should not be interpreted as a proposal to standardize this language for EAR approaches.

```
conceptual-schema            = "CONCEPTUAL SCHEMA"
                                    conceptual-schema-name
                               entity-type-description
                                    {entity-type-description}
                               relationship-type-description
                                    {relationship-type-description}.

entity-type-description      = "ENTITY-TYPE" entity-type-name
                               "IDENTIFIER" identifier
                               "DESCRIPTION" attribute-type-description
                                    {attribute-type-description}.

relationship-type-description =
                               "RELATIONSHIP-TYPE"
                                    relationship-type-name
                               "DIMENSION" unsigned-integer
                               "COLLECTION" entity-type-name
                                    {entity-type-name}
                               "CARDINALITY"
                                    entity-type-name minc "," maxc
                                    {entity-type-name minc "," maxc}
                               {"FUNCTIONAL DEPENDENCY"
                                    entity-type-name {entity-type-name}
                                    "ON" entity-type-name}
                               ["IDENTIFIER" identifier]
                               ["DESCRIPTION"
                                    attribute-type-description
                                    {attribute-type-description}].
```

```
attribute-type-description  = attribute-type-name.

minc                        = unsigned-integer | letter.

maxc                        = unsigned-integer | letter.

conceptual-schema-name      = identifying-name.

entity-type-name            = identifying-name.

identifier                  = attribute-type-name
                                {"," attribute-type-name}.

attribute-type-name         = identifying-name.

relationship-type-name      = identifying-name.

identifying-name            = letter {letter | digit | hyphen}.
```

The letter, digit, hyphen, and unsigned-integer are not further defined here.

The identifying-names must be unique within the conceptual schema. The other semantics should be clear from descriptions in section D.2 and D.3.

D.4. GRAPHIC FORMALISM.

Structure diagrams of entity-types and relationship-types may be drawn in the EAR approaches, using the following symbols:

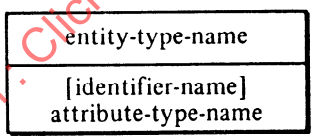


Figure D.11. Symbol for an entity-type in an EAR structure diagram.

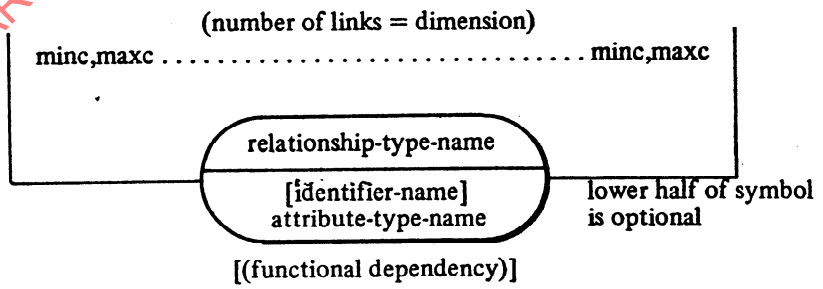


Figure D.12. Symbol for a relationship-type in an EAR structure diagram.

D.5. MODELLING.

Many modelling techniques have been proposed in accordance with EAR approaches. These techniques can be classified into two categories:

In modelling directly the universe of discourse entities and relationships are identified in the propositions of the universe of discourse. Chen [3] proposes such a modelling technique by showing an analogy between: subject and entity; verb and relationship.

Modelling in two steps consists of:

- identifying attributes, i.e. any information taking values;
- constructing entity-types and relationship-types by analysing attribute-types.

The second step may itself be considered in two ways: Some authors [8, 9] give algorithms allowing automatic construction of entity-types and relationship-types with an exhaustive list of attribute-types and functional dependencies. Other authors [5] propose interactive automata allowing designers to propose a structure, formally checking by reformulating the structure and being asked if it is correct.

D.5.1. SOME PRAGMATIC MODELLING RULES.

In the EAR approaches relationships are defined among entities, not among attributes; hence, the choice of entities is important. Some pragmatic rules have been used to decide how to handle such problems:

1. If an entity-type has but one relevant attribute-type, it probably should be classified as an attribute-type of some other entity-type.
2. If attributes of several entities refer to the same entity, that entity should be classified in an entity-type in its own right.

Explanation of these pragmatic rules illustrates that the choices must be made by considering the universe of discourse and not only its description.

The decision as to what is an attribute-type and what is an entity-type cannot usually be decided a priori. The EAR approach is iterative, and what appears to be an attribute at an early stage of the modelling process may turn out to be an entity and vice versa.

Relationships are not entities. However, it is possible to construct an entity which is the transformation of a relationship, as in - for example - transforming the association between a person and his boat via the relationship OWNS over the entity-types PERSON and BOAT into a relationship OWNS over the entity-types PERSON, BOAT, and OWNERSHIP. This may be decided for several reasons. One reason is to give the concept, originally modelled as the relationship-type, an identifier and not to use the identifiers of the collection of entities. Another is to provide what the designer feels is a more convenient view of the universe of discourse.

D.5.2. FORMAL RULES FOR MODELLING.

Although we have discussed some pragmatic rules which are appropriate for modelling, formal rules also exist. We give here only a short summary. Detailed descriptions can be found in e.g. [9].

Verification:
Verification allows us to insure that in every occurrence of an entity-type or relationship-type only one value of any given attribute-type is included.

Normalization:
Normalization in the EAR approaches insures that every attribute of a relationship cannot be verified upon a subset of the identifier of the relationship.

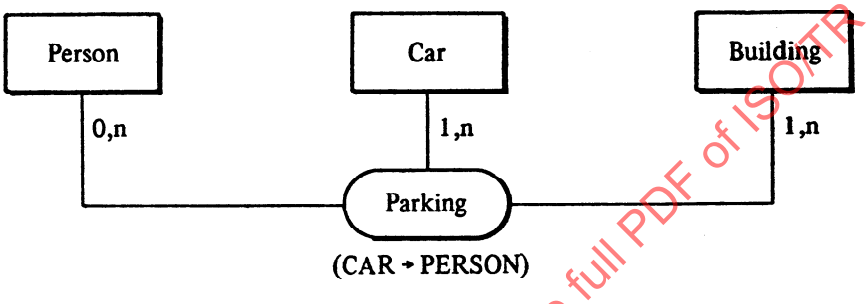


Figure D.13. Ternary relationship with functional dependency.

Decomposition:
Decomposition in the EAR approaches allows splitting a relationship-type of dimension n into several relationship-type with smaller dimensions without loss of semantics, provided functional dependencies defined over the relationship-type are used. It is necessary to verify that the occurrences used in the common part are the same. For example, in figure D.13, in the relationship-type PARKING defined over the entity-types PERSON, CAR, and BUILDING the functional dependency (CAR -> PERSON) - a car must belong to at most one person - may be defined. This situation may be decomposed into the situation of figure D.14:

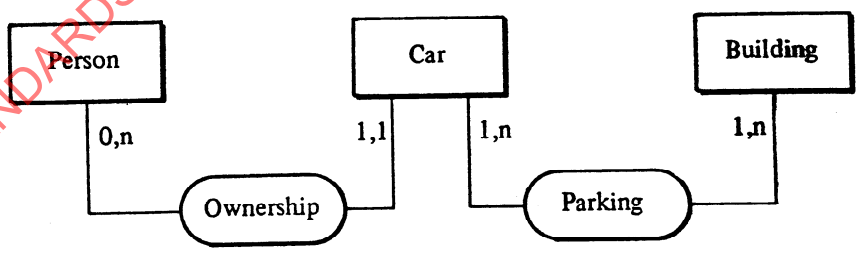


Figure D.14. Decomposition of the example of figure D.13.

Note, that this decomposition does not "objectify" the relationship-type as was done in the example of figure D.10.

D.6. EXAMPLE CONCEPTUAL SCHEMA.

D.6.1. GRAPHIC REPRESENTATION.

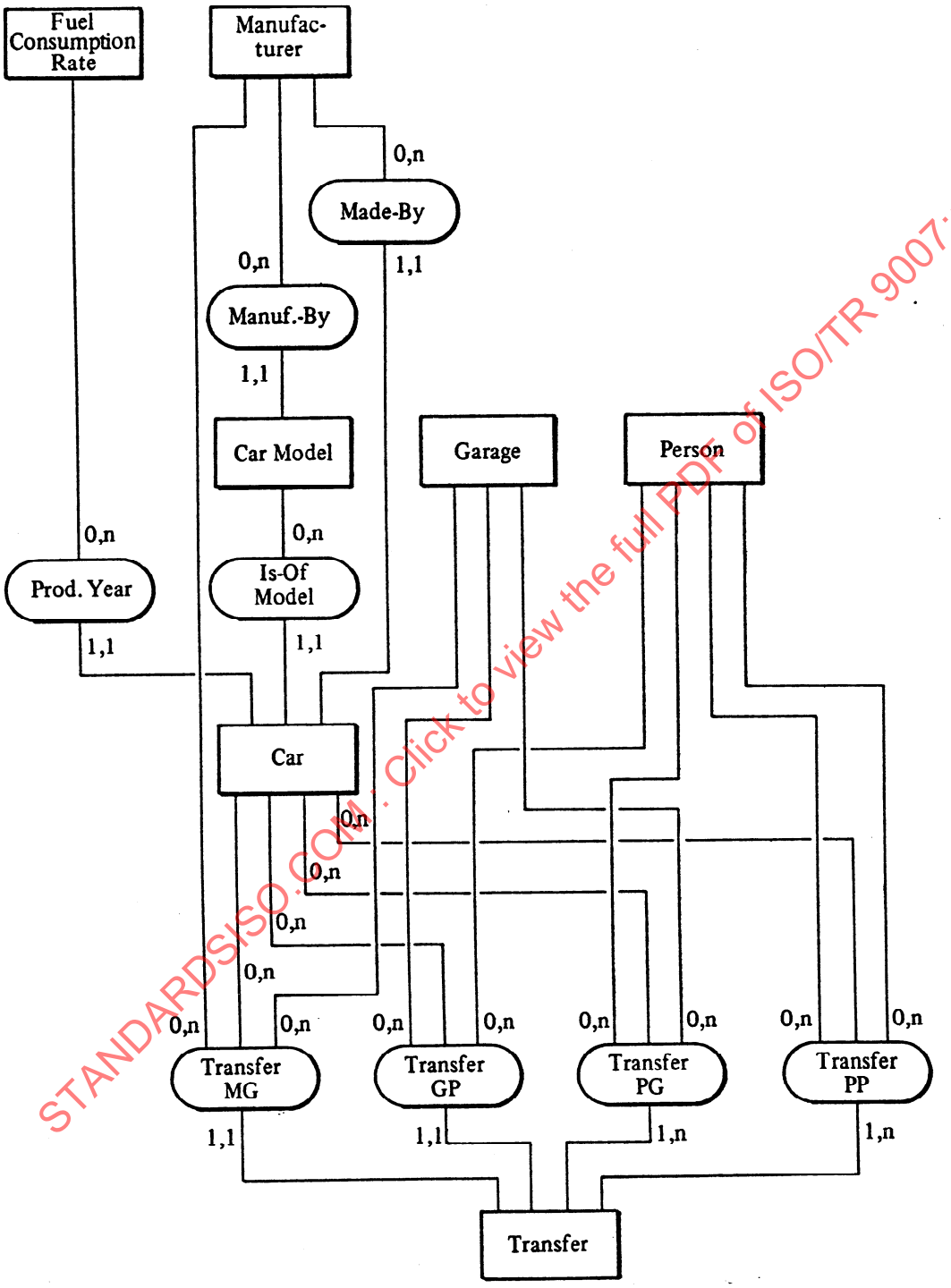


Figure D.15. Example conceptual schema.

D.6.2. LANGUAGE EXAMPLE.

A description of the conceptual schema in the language of the grammar defined in section D.3, is as follows:

```
CONCEPTUAL SCHEMA    car-registration

ENTITY-TYPE          manufacturer
IDENTIFIER           manuf-id
DESCRIPTION          manuf-id
                     is-operating

ENTITY-TYPE          car-model
IDENTIFIER           model-id
DESCRIPTION          model-id
                     fuel-cons-spec

ENTITY-TYPE          car
IDENTIFIER           reg-no
DESCRIPTION          reg-no
                     serial-no
                     destroyed-date

ENTITY-TYPE          fuel-consumption-rate
IDENTIFIER           year-id
DESCRIPTION          year-id
                     max-cons

ENTITY-TYPE          garage
IDENTIFIER           garage-id
DESCRIPTION          garage-id
                     is-trading

ENTITY-TYPE          person
IDENTIFIER           person-id
DESCRIPTION          person-id

ENTITY-TYPE          transfer
IDENTIFIER           transfer-car, transfer-date, seq-no
DESCRIPTION          transfer-car
                     transfer-date
                     seq-no

RELATIONSHIP-TYPE    manuf-by
DIMENSION            2
COLLECTION           manufacturer
                     car-model
CARDINALITY          manufacturer 0,n
                     car-model    1,1

RELATIONSHIP-TYPE    made-by
DIMENSION            2
COLLECTION           manufacturer
                     car
CARDINALITY          manufacturer 0,n
                     car          1,1
```

```
RELATIONSHIP-TYPE   is-of-model
DIMENSION           2
COLLECTION          car-model
                    car
CARDINALITY         car-model 0,n
                    car       1,1

RELATIONSHIP-TYPE   prod-year
DIMENSION           2
COLLECTION          fuel-consumption-rate
                    car
CARDINALITY         fuel-consumption-rate 0,n
                    car                   1,1

RELATIONSHIP-TYPE   transfer-mg
DIMENSION           4
COLLECTION          car
                    transfer
                    manufacturer
                    garage
CARDINALITY         car          0,n
                    transfer     1,1
                    manufacturer 0,n
                    garage       0,n

RELATIONSHIP-TYPE   transfer-gp
DIMENSION           4
COLLECTION          car
                    transfer
                    garage
                    person
CARDINALITY         car      0,n
                    transfer 1,n
                    garage   0,n
                    person   0,n

RELATIONSHIP-TYPE   transfer-pg
DIMENSION           4
COLLECTION          car
                    transfer
                    person
                    garage
CARDINALITY         car      0,n
                    transfer 1,n
                    person   0,n
                    garage   0,n

RELATIONSHIP-TYPE   transfer-pp
DIMENSION           4
COLLECTION          car
                    transfer
                    person
CARDINALITY         car      0,n
                    transfer 1,n
                    person   0,n
                    person   0,n
```

D.7. CHECK LIST FOR THE CONCEPTUAL SCHEMA.

The following analysis illustrates whether or not the necessary propositions about the universe of discourse are captured in the conceptual schema. An E implies that the assertion is described in the model. An * refers to remarks at the end of this section.

CHECK		NECESSARY PROPOSITIONS
E	1.	The universe of discourse to be described has to do with the registration of cars and is limited to the scope of interest of the Registration Authority.
E	2.	Each car manufacturer has a unique name.
*	3.	New car manufacturers can start operation provided they have the permission of the Registration Authority.
*	4.	The Registration Authority cannot withdraw the permission.
*	5.	At any time not more than five autonomous manufacturers may operate.
*	6.	Manufacturers may cease to operate, provided they do not own cars anymore.
E	7.	Each car manufacturer constructs cars in several models.
E	8.	A car is of a particular model.
*	9.	A manufacturer gives a serial number to each car he produces.
*	10.	This serial number is unique for all cars of one manufacturer.
*	11.	A newly produced car is registered by the Registration Authority as soon as practicable.
(E)	12.	At this time the car is registered as belonging to the manufacturer which produced it. Therefore the first owner will be the manufacturer who produced the car.
*	13.	Only the Registration Authority will assign a registration number to each registered car.
(E)	14.	This registration number is unique for all cars for all time.
E	15.	A car has a year of production.
*	16.	Only in January may a car be registered as being produced in the previous year.
E	17.	Cars can be destroyed whereupon the date of destruction is recorded.

* 18. The car's history has to be kept until the end of the second subsequent calendar year after its destruction. Thereafter it is removed from the registered information.

(E) 19. The name of the car model is unique for all car models for all time.

* 20. Any specific car model is constructed by only one manufacturer.

* 21. From time to time new models will be introduced.

E 22. All cars of the same car model have the same fuel consumption.

(E) 23. This fuel consumption must be known to the Registration Authority.

* 24. The fuel consumption of a car will be between 4 and 25 litres per 100 km.

* 25. The fuel consumption averaged over all individual cars produced by a particular manufacturer in a particular year is required not to exceed a maximum value which is the same for each manufacturer.

E 26. The maximum fuel consumption rate may change from year to year.

* 27. At the end of January a message is sent to a manufacturer who has failed to meet this requirement in the previous year.

E 28. Each garage has a unique name.

* 29. New garages may be established.

E 30. Garages may own cars.

* 31. A garage must not have, at any time, cars registered as belonging to the garage, from more than three manufacturers (which three does not matter, and for a particular garage may vary with time).

* 32. An existing garage may be closed down, provided it does not have any cars registered to it.

E 33. A particular person may have one or more cars registered as belonging to him or her.

E 34. It is also possible for two or more people to have one or several cars registered as belonging to them jointly and simultaneously.

E 35. People have unique names.

* 36. People are only known to the Registration Authority if they own or have owned (one or more) cars, which still are known to the Registration Authority.

* 37. At any time a car is owned by either

- its manufacturer,
- a garage,
- a person,
- a group of persons,

but not jointly by two or more of these categories.

E 38. Transfer of ownership is registered including the date of transfer, the previous owner(s) and the new owner(s).

* 39. Transfer of ownership cannot take place anymore after a car's destruction.

* 40. However, transfer of ownership may be recorded after the car's destruction, provided the transfer of ownership took place before the car's destruction.

E 41. Each manufacturer distributes new cars to several independent garages, each which may receive cars from more than one manufacturer.

(E) 42. Therefore a garage always will be a car's second owner.

E 43. Manufacturers do not distribute cars to other manufacturers or directly to people.

E 44. Each garage may sell - i.e., cause transfer of registered ownership of - new or used cars to people, and may buy - i.e., cause transfer of registrered ownership of - cars from people.

E 45. Garages are not allowed to sell cars to other garages.

E 46. Garages are not allowed to sell cars to manufacturers.

E 47. People can sell cars to other people or buy cars from other people.

Remarks for the EAR approaches:

* A check between parenthesis means, that the conceptual schema provides a description of the assertion in the information base, but that the assertion cannot be enforced as a rule. For example, check 12, 23, and 42. Note, that the uniqueness constraints in checks 14 and 19 are supported, but not for all time.

* No authorization rules are included, for example, checks 3, 4, 9, 11, and 13.

* Validation rules, although being static rules, are not included. For example, checks 5, 10, 24, 25, 31, 36, and 39.

* Exclusiveness of relationships is not included. For example, check 37.

* No dynamic rules or constraints are included, therefore checks 6, 20, and 32 are not applicable. Note, that the static constraint "any specific car model is produced by one manufacturer at a time" is not meant in check 20.

* Prescriptive rules for interactions are not part of the conceptual schema. For example, checks 16, 18, 21, 27, 29, and 40.

D.8. MAPPING OF AN EAR CONCEPTUAL SCHEMA TO A NETWORK DATA BASE SCHEMA AND A RELATIONAL DATA BASE SCHEMA.

A procedure for converting an EAR conceptual schema first to a network data base schema [10] and then to a relational data base schema [11] can roughly be outlined as follows:

1) Put the attribute-types of a single entity-type together into a record-type.
2) Take the one-to-many relationships-types and convert them into set-types.
3) Promote the many-to-many, the n-ary (n>2) relationship-types and the relationship-types, that have attribute-types, to record-types and promote the connecting lines into set-types.

This gives a data structure diagram with information-bearing sets. In our example it yields the following diagram:

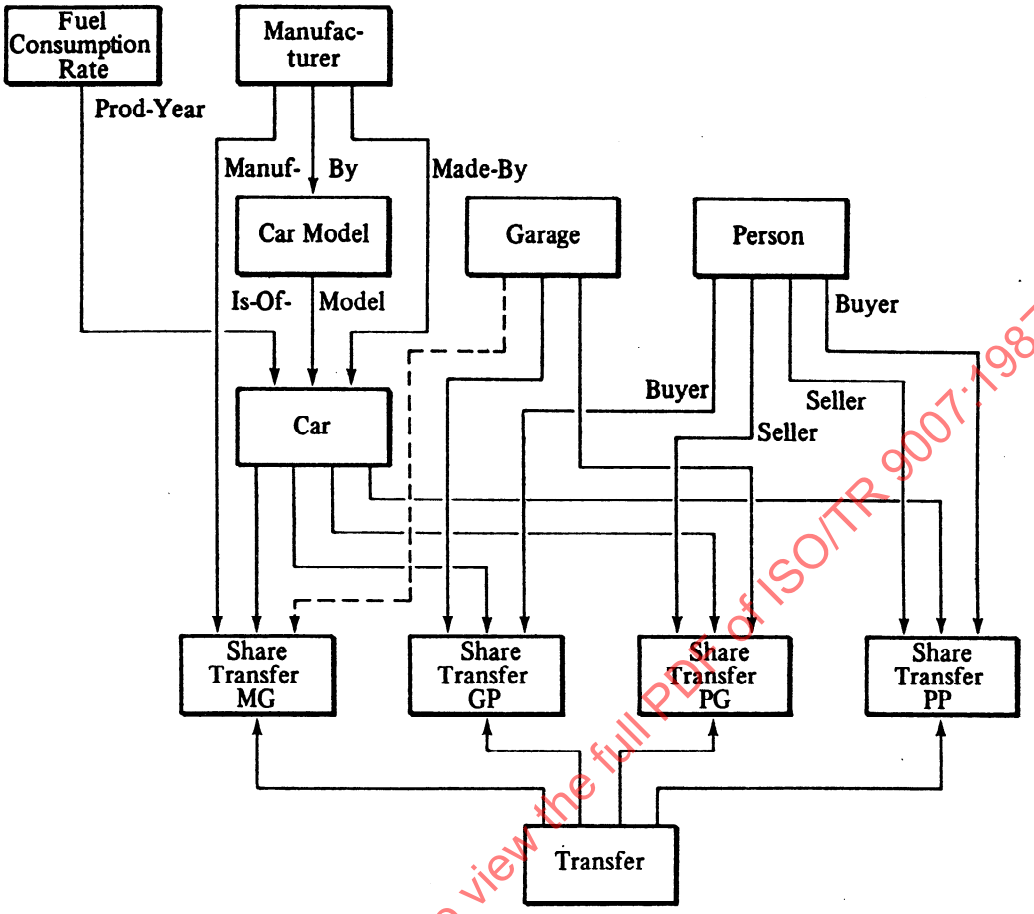


Figure D.16. Example Data Base schema.

To get a relational data base schema we map the record-types into relational data tuples representing the sets by "owner-keys" referring to "owner-tuples" (e.g. manuf-by in CAR-MODEL):

MANUFACTURER (manuf-id, is-operating)
CAR-MODEL (model-id, manuf-by, fuel-cons-spec)
CAR (reg-no, made-by, serial-no, is-of-model, prod-year, destroyed-date)
FUEL-CONSUMPTION-RATE (year-id, max-cons)
GARAGE (garage-id, is-trading)
PERSON (person-id)
TRANSFER (transfer-car, transfer-date, seq-no)
SHARE-TRANSFER-MG (transfer-car, transfer-date, seq-no,
selling-manuf, buying-garage)
SHARE-TRANSFER-GP (transfer-car, transfer-date, seq-no,
selling-garage, buyer)
SHARE-TRANSFER-PG (transfer-car, transfer-date, seq-no,
selller, buying-garage)
SHARE-TRANSFER-PP (transfer-car, transfer-date, seq-no, seller, buyer)

D.9. REFERENCES.

[1] BACHMAN, C.W. ´Data Structure Diagrams`,
In: Data Base 1, No.2, 1969, (Publication of ACM Special Interest Group on Business Data Processing).

[2] ENGLES, R.W. ´A tutorial on data base organisation`,
In: Annual Review in Automatic Programming, Vol 7, Part 1, Pergamon Press, 1972.

[3] CHEN, P.P. ´The entity - Relationship model - Toward a Unified View of Data`,
In: ACM TODS, Vol 1, No. 1, 1976.

[4] MOULIN, P., RANDON, J. and TARDIEU, H.
´Conceptual model as a data base design tool`,
In: Modelling in data base management systems, proceedings IFIP TC2 Conference, Freudenstadt 1976; North-Holland Publishing Company.

[5] TARDIEU, H., HECKENROTH, H., PASCOT, D., and NANCI, D.
´A method, a formalism, and tools for data base design: Three years of experimental practice`,
In: Chen, P.P. (ed.), Entity-Relationship Approach to System and Analysis, North-Holland Publishing Company, 1980.

[6] OPEN UNIVERSITY ´Block II, A. Conceptual Modelling, B. Logical Modelling`,
In: Course M352 Computer-based Information Systems, Open University Press, 1980.

[7] PALMER, I.R. and ROCK-EVANS, R.,
´Data Analysis`,
In: Computer Weekly, 1980.

[8] FLORY, A. ´An approach to design an entity relationship scheme`,
In: Chen, P.P. (ed.), Entity-Relationship Approach to System and Analysis, North-Holland Publishing Company, 1980.

[9] MIJARES, F. and PEEBLES, R.
´A methodology for the design of logical data base structures`,
In: Modelling in data base management systems, proceedings IFIP TC2 Conference, Freudenstadt 1976; North-Holland Publishing Company.

[10] CODASYL DBTG ´January 78 Report`,
CODASYL JOD, 1978.

[11] CODD, E.F. ´A relational model of Data for Large Shared Data Banks`,
In: Communications of the ACM, Vol. 13, No.6, 1970.

APPENDIX E. THE BINARY RELATIONSHIP APPROACHES.

E.1. EMPHASIS OF THE APPROACHES.

The Binary Relationship approaches to conceptual schema definition have their origin in the work of authors such as Abrial in the early seventies, and Bracchi and Senko in the mid seventies. During this period a few implementations of binary relationship systems were also attempted. More recently fresh implementations of binary relationship systems have sought to demonstrate that application programming and end user interfaces to data base management systems can be realized at a level close to the conceptual schema.

In this chapter, we present a synthesis of the Binary Relationship approaches, which is not the product of any single author in particular. Other approaches to the binary model exist in the literature. What all these approaches have in common, is a representation of information "elements" by means of instances of binary associations, i.e. sentences in which only two terms play a role.

A common feature of Binary Relationship approaches is the use of graphic notation to illustrate parts of the conceptual schema and the information base.

An information base is an attempt to describe a part of the universe of discourse [1]. From a modelling viewpoint, this universe of discourse presents itself at each moment of time as a set of entity and binary association occurrences [2]. The population and semantics of the universe of discourse must be modelled in a way that agrees with the existing propositions about those occurrences.

Remark: The notions of population, occurrence and type above and below are here applied to concepts of object and binary association, as done commonly by proponents of this method such as Abrial [3], Bracchi [4], Senko [5], as well as Durchholz [6] and several others.

The modelling process occurs in two phases: On one side one has the universe of discourse, on the other the information base and conceptual schema.

The modeller, or analyst, performs three tasks: he names (refers to) or establishes naming conventions for the entity which he observes in the universe of discourse, he classifies it (which is also a naming action of a kind) and prepares a conceptual schema that will describe this classification as well as allows populating it with a name of or reference to the observed entity.

In the Binary Relationship approaches, it is taken as a starting point that the conceptual schema's function is mainly to introduce classification and order in the description of the universe of discourse, not just be a "flat" exhaustive description on the occurrence level. The modelling of the universe of discourse on this occurrence level may be considered sufficient from a theoretical point of view by certain other approaches (since it can conceivably achieve any desired degree of completeness). That kind of modelling method, however, lacks many pragmatical and practically useful properties which, for instance, are a consequence of the type concept.

The Binary Relationship approaches, as presented here, fully implement the distinction between lexical and non-lexical objects as outlined in section 2.1 of chapter 2 as a means to resolve some of the confusion between things and the names of things that occur in so many information descriptions and information exchanges. At the same time it provides a systematic treatment of such natural concepts like type.

These Binary Relationship approaches do not themselves treat the language in which a universe of discourse is described in any other way than to emphasize its existence and necessity. It is implicitly assumed as a postulate that there is semantical equivalence in the sense of chapter 3, section 3.4, between:

* the propositions about the universe of discourse,

* the sentences describing them in the conceptual schema and information base (not necessarily in a one-to-one correspondence).

Considering the objectives for a conceptual schema, cited in section 1.9 of chapter 1, the following remarks could be made about these approaches:

1. "To provide a common basis for understanding the general behaviour of the universe of discourse."
   The Binary Relationship approaches usually provide a formal language to completely describe the conceptual schema, and a graphic formalism or notation covering major aspects of the conceptual schema. The use of the basic concepts - entity, entity-name, binary relationship and constraint - makes it possible to completely describe the dynamic rules and consstraints of the conceptual schema and information base, as well as the static rules and constraints (This characteristic is also available in the IPL approaches). The fundamental distinction between things and their names is a great help in the ease of understanding and teaching [5, 7]. The challenging part in the teaching and understanding is in the area of constraints.

2. "To define the allowed evolution and manipulation of the information about the universe of discourse."
   In the Binary Relationship approaches the basic concepts are atomic. As a consequence any given constraint is explicitly formulated. Therefore changes in the universe of discourse can be absorbed by just adding or deleting, but not changing, the types of binary relationships, entities or entity names, or by just changing constraints. This results in a more relative stability as opposed to approaches where some constraints are "packed" into the "basic" concepts.

3. "To provide a basis for interpretation of external and internal syntactical forms which represent the information about the universe of discourse."
   To serve this role it is needed that a conceptual schema contains the deepest structure of the problem description in semantically irreducible elements. The distinction between things and their names present in these Binary Relationship approaches provides a deep structural aspect while the bi-

nary relationships and the subtyping capability provide the semantical irreducibility.

4. "To provide a basis of mappings between and among external and internal schemata."
The maximum degree of stability for the purpose meant here is reached with semantically irreducible constructs and constructs free of additional constraints. This is the case in the Binary Relationship approaches.

E.2. PRIMITIVE CONCEPTS OF THE APPROACHES.

When recognizing the distinction between lexical and non-lexical objects and their disparate functions, the concept of type is needed. Indeed, one or several lexical objects will specify one or several non-lexical objects corresponding to entities in the universe of discourse. Let us take a specification, for example:

´Rita` (lexical object)

We now want to express that we talk about a certain entity in the universe of discourse corresponding to the string ´Rita`. To denote entities such as in this case a girl-entity, we use the notation <girl>. So ´Rita` refers to a <girl> entity. Suppose this <girl> is referred to in our language by the word ´girl`. An obvious construct would be then to talk of:

´girl` called ´Rita`

whereby ´girl` is introduced as a lexical object (or name) for the entity-type of <girl>. This lexical object ´girl` is therefore a lexical object at the conceptual schema level.

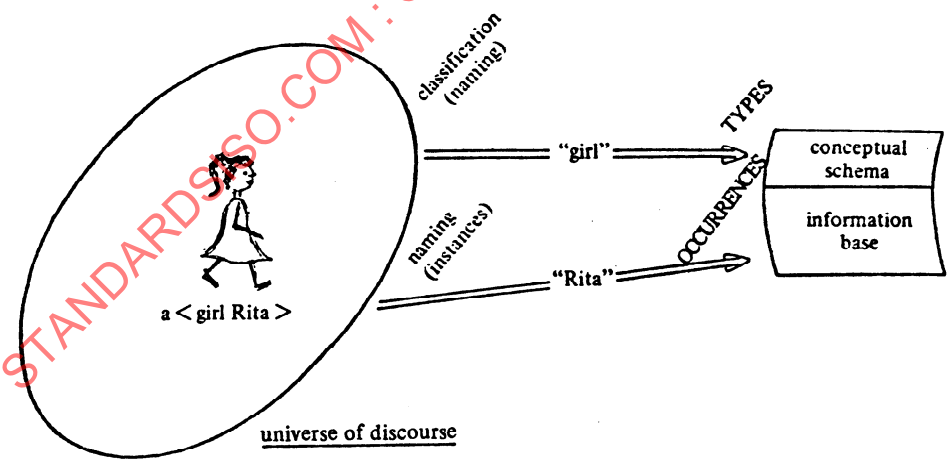


Figure E.1. Classifying and naming the <girl Rita>.

The non-lexical object named by ´girl` is a class of entities <girl>s in the universe of discourse.

The same example we can now elaborate a little further and establish also a lexical object or name ´girlname` for the class of all entities in the universe of

discourse that are names of <girl>s. So ´Rita` is an occurrence of ´girlname`. Note that again ´girlname` is a lexical object at the conceptual schema level.

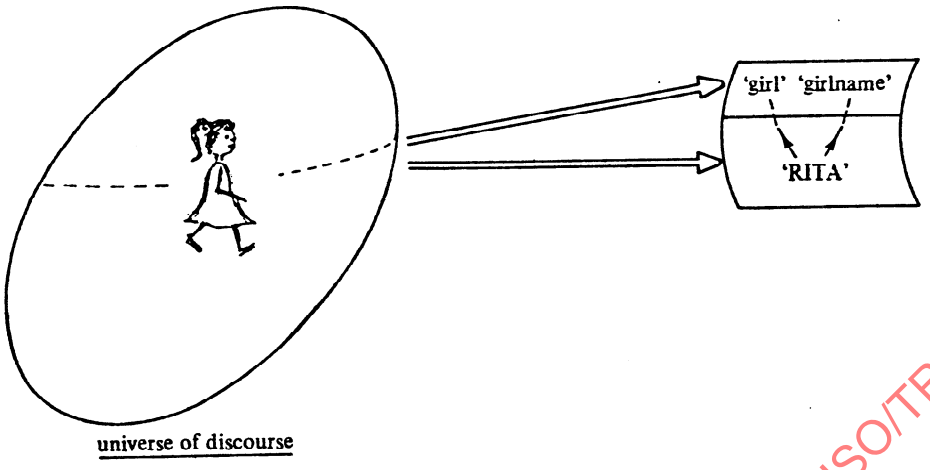


Figure E.2. Typing and naming in the universe of discourse.

Observe also that, when described and entered in the conceptual schema, this step will precisely model the "naming"-arrow in figure E.2. We see that ´girlname` denotes another kind of type of lexical objects. A link between the two types above, one lexical and one non-lexical, is already visible in the word "called". In the Binary Relationship approaches discussed here this type of link is named a bridge (Nijssen [8]).

It must be mentioned already here that these Binary Relationship approaches do not prohibit synonyms or homonyms to be used as names or references. However, for the sake of determinism, it is required that every entity-image or object occurrence in the conceptual schema and information base possesses, at all times, at least one way of unambiguously referring to it.

Summarizing, we talk of

´girlname`

as specifying a type of lexical objects of which ´Rita` is an individual occurrence, and of

´girl`

as referring to a type of non-lexical objects of which

´girl` called ´Rita`

is an individual occurrence.

Formally, we shall have thus:

1. - Linguistic objects:

   1.1. lexical objects, corresponding to individual "utterable" entities. Example: ´Jones`, ´Rita`, ´girl`.

1.2. Non-lexical objects, corresponding to "non-utterable" entities. Non-lexical objects specify entities by using lexical objects (in a manner elaborated below). Example: the ´person` called ´Jones`, the ´girl` called ´Rita`. Note, that ´person` and ´girl` are used here as references to types rather than to instances, as in 1.1.

2. - Binary relationship instances (between two linguistic objects):

2.1. Idea, between two non-lexical objects.

2.2. Bridge, between a lexical and a non-lexical object.

2.3. Phrase, between two lexical objects.

3. - Linguistic object types:

3.1. Lexical object types (LOTs).

3.2. Non-lexical object types (NOLOTs).

This classification into types is an essential part of the modelling process (Falkenberg [7]). It depends usually and mostly on the syntax and semantics of the translation of the propositions in the universe of discourse into the language of the information system. It also depends on the modeller´s awareness of the shortcomings, abstractions, interpretations, and naming conventions introduced while making this translation.

Further corresponding to these types of linguistic objects there are types of ideas, bridges and phrases:

4. - Binary relationship types:

4.1. Idea types, conveying the "real" information between non-lexical objects.

4.2. Bridge types, linking the naming lexical-objects to the named non-lexical objects.

4.3. Phrase types, carrying relations between lexical objects only.

And, also having types of linguistic objects, we shall want to "subclassify" them into subtypes, depending on the distinction between shared and non-shared binary relationship types on linguistic object types on one hand, and pure "conceptual" subclassification of the (future) populations of those linguistic object types on the other hand.

For example, we might consider the subtypes ´man`, ´woman`, ´employee` of the type ´person` mentioned above. Almost always a subtype is introduced because of the existence of particular distinguishing ideas on the "supertype". In this example, ´man` might be characterized by a military record, ´woman` by natural children, ´employee` by a firm, and so on. Hierarchies of subtypes are of course perfectly possible and permitted in the Binary Relationship approaches. Only subtyping between a lexical and a non-lexical object type is not allowed

for obvious reasons. It is important to note that subtypes need not be disjoint.

So we have:

5. - Subtypes:

5.1. Non-lexical subtypes, i.e. which induce subsets of non-lexical object types.

5.2. Lexical subtypes, i.e. which induce subsets of lexical object types.

Finally, realizing some propositions of the universe of discourse, or more precisely, some necessary propositions, are about propositions themselves and therefore translated into predicates about linguistic objects and binary relationship instances, we need to express these in static and dynamic constraints (at the type level). Since the existence of those propositions has probably influenced the shape of the universe of discourse the constraints will also influence the modelling process.

So we end here with:

6. - Constraints, both static and dynamic.

All operators, from the vantage point of the conceptual schema, work on the "occurrence" or "instance" level. There are actually few of them.

A very important axiom, which is introduced to make these operators generally meaningful, is the following:

> In every state of the information base, every linguistic object instance must be an occurrence of some linguistic object type; every binary relationship instance is an occurrence of some binary relationship type; and most important, every linguistic object possesses a way of uniquely referring to it by means of one or more lexical objects.

The operators defined for this example of the Binary Relationship approaches are:

7. - Operators:

7.1. list (... a binary relationship instance),

7.2. add (... a binary relationship instance),

7.3. delete (... a binary relationship instance),

7.4. qualify (... an linguistic object as belonging to a subtype),

7.5. unqualify (... an linguistic object from out of a subtype),

7.6. equate (... identify two different linguistic objects as meaning or referring to one and the same thing).

7.7. together do (... perform a sequence of operators but decide on its permissibility on the basis of the sequence as a whole).

The semantics of these elementary operators are directly interpretable in terms of chapter 2 in the cases (7.1) - retrieval, (7.2) - insertion, (7.3) - deletion, and (7.7) - permissible action. The same or similar operators exist in most approaches. The operators (7.4) - qualify, (7.5) - unqualify, and (7.6) - equate, however, are typical for this example of the Binary Relationship approaches. The operators "qualify" and "unqualify" derive from the notion of subtypes:

qualify — takes a reference to a linguistic object and transforms it to a reference to a linguistic object in one of the subtypes of the linguistic object type containing the linguistic object.
For example, "qualify the manufacturer `Ford` as operating-manufacturer".

unqualify — does the opposite; it removes the ability to use the reference for linguistic objects of the subtype concerned.
For example, "unqualify the manufacturer `PSC` as operating-manufacturer".

The operator "equate" is particular since it is a consequence of how these Binary Relationship approaches implement the distinction between names and things:

equate — establishes two different terms as being references to the same entity in the universe of discourse, thus needing "non-lexical" identification while retaining all existing naming conventions for it.
For example, "equate (the manufacturer called Ford,the manufacturer of the car-model Mustang)".

E.3. GRAMMAR AND SEMANTICS.

E.3.1. THE LANGUAGE AND ITS RELATION TO THE UNIVERSE OF DISCOURSE.

Referring to chapter 3, section 3.3, we may repeat that certain general concepts are primitive notions that characterize the grammar of any language. They are:

TERM
A linguistic object that refers to an entity;

SENTENCE
A linguistic object which expresses a proposition;

FUNCTOR
A linguistic object that refers to a function on other linguistic objects taking as arguments (input) a list of linguistic objects (terms, sentences, functors) and yielding as a value (output) a single, uniquely determined linguistic object (term, sentence, functor).

The functors used for the Binary Relationship approaches are the usual ones of predicate, operator, quantifier, etc, and will be made explicit along the way.

It is possible to give an interpretation for these concepts in the language used for conceptual schema definition in these Binary Relationship approaches as follows:

TERMS:

a) ELEMENTARY TERMS: Linguistic Object Type Names, Binary Relationship Type Names, Role Names, Individual Names.

Example: ´Rita`, ´girl`;

b) DESCRIPTIVE TERMS: References linking one or more lexical objects (elementary terms) to a non-lexical object.

Example: ´The company employing Rita`.

SENTENCES:

a) ELEMENTARY SENTENCES: Declarations of Linguistic Object Types, Binary Relationship Types.

b) FUNCTORIAL SENTENCES: Obtained through application of functors (mainly predicates) to TERMS and ELEMENTARY SENTENCES and used for constraint definition.

Examples follow in section E.6.

Elementary sentences (declarations) are built up mainly with the primitive predicate "is-a" in some equivalent form (namely the "add" of a "bridge" instance). Note that this predicate connects a term from the conceptual schema level with one from the meta-conceptual level, containing such names as ´OBJECT TYPE` and ´IDEA TYPE`. Other primitive predicates in the model are "is-a-subset-of", with the obvious meaning, and "has-roles" and "connects-with" which are used in declaring roles for idea, bridge, and phrase types below.

In a practical application, one would probably find also useful certain primitive or (semi-primitive) predefined predicates specifying a logical representation for lexical object types, but we shall not consider those here to avoid cluttering up the real issues.

The example language of section E.6 is to be considered ad hoc for the purposes of this chapter. No implied suggestion for any kind of standard is implied.

The grammar of the language used for the example is based on the following principle:

> Since the grammar describes conceptual schemata, it can itself be considered as a language in which we model a particular universe of discourse, namely the universe of discourse of the conceptual schemata of the Binary Relationship approaches. Therefore a "declaration" of a lexical object type, for instance, is nothing but the "add" of a bridge connecting the meta-schema-LOT ´LEXICAL-OBJECT-TYPE-NAME` or ´LOTNAME` with the meta-schema NOLOT ´LEXICAL-OBJECT-TYPE` or ´LOT`.
>
> Example:
>
> add NOLOT called ´GARAGE´
>
> declares a NOLOT with NOLOTNAME ´GARAGE´, by inserting this piece of information as a binary (bridge-) relationship instance in the conceptual schema.

E.3.2. FORMAL SYNTAX.

The formal syntax of the language used in the example (section E.6) is given below, using the syntax notation of appendix C:

```
(R1)  conceptual-schema      = "begin"
                               "CONCEPTUAL-SCHEMA called" schema-name ";"
                               nolot-declaration {nolot-declaration}
                               lot-declaration {lot-declaration}
                               {subtype-declaration}
                               {idea-declaration}
                               bridge-declaration {bridge-declaration}
                               {phrase-declaration}
                               {constraint-declaration}
                               "end".

(R2)  nolot-declaration      = "NOLOT called" (nolot-name | nolot-name-list).

(R3)  lot-declaration        = "LOT called" (lotname | lotname-list).

(R4)  subtype declaration    = nolot-subtype | lot-subtype.

(R5)  nolot-subtype          = "NOLOT called"
                                     (nolot-name-1 | nolot-name-1-list)
                               "is subtype-of NOLOT called" nolot-name-2 ";".

(R6)  lot-subtype            = "LOT called" (lot-name-1 | lotname-1-list)
                               "is subtype-of LOT called" lot-name-2 ";".
```

```
(R7)  idea-declaration        = "IDEA (with-first ROLE (called" role-name-1
                                      "and on NOLOT called" nolot-name-1 ")"
                                "and with-second ROLE (called" role-name-2
                                      "and on NOLOT called" nolot-name-2 "))"
                                "is called" idea-name ";".

(R8)  bridge-declaration      = "BRIDGE (with-first ROLE (called" role-name-1
                                      "and on NOLOT called" nolot-name ")"
                                "and with-second ROLE (called" role-name-2
                                      "and on LOT called" lot-name "))"
                                "is called" bridge-name ";".

(R9)  phrase-declaration      = "PHRASE (with-first ROLE (called" role-name-1
                                      "and on LOT called" lot-name-1 ")"
                                "and with-second ROLE (called" role-name-2
                                      "and on LOT called" lot-name-2 "))"
                                "is called" phrase-name ";".

(R10) constraint-declaration = "CONSTRAINT called" constraint-name
                                "is declared as" sentence ";".

(R11) nolot-name-list         = "{" nolot-name {nolot-name} "}" ";"

(R12) lotname-list            = "{" lot-name {lot-name} "}" ";"

Remark:  nolot-name-1-list and lot-name-1-list follow the same syntax.

(R13) schema-name             = identifying-name.

(R14) nolot-name              = identifying-name.

(R15) lot-name                = identifying-name.

(R16) role-name               = identifying-name.

Remark:  nolot-name-1, etc, follow the same syntax.

(R17) idea-name               = identifying-name.

(R18) bridge-name             = identifying-name.

(R19) phrase-name             = identifying-name.

(R20) identifying-name        = "'" letter {letter | digit | hyphen} "'".
```

Remark: The syntax of the constraint sentences, as well as the syntax of letter, digit and hyphen are not further defined here.

This particular syntax was taken, slightly rephrased and simplified, from the RIDL report (Meersman - Vermeir [9]).

It should be clear that a conceptual schema definition such as above can be "stored" by considering each "declaration" as a sentence to be added. In general delete, list, etc, of course have also their usual meaning in this context. An example of such a "session" is to be found in section E.6.

Example:

```
add CONCEPTUAL-SCHEMA called ´CAR-REGISTRATION´;
add NOLOT called {´CAR´ ´CAR-MODEL´ ´DATE´ ´TRANSFER´ ´OWNER´};
add LOT called {´REG-NO´ ´SERIAL-NO´};
add NOLOT called ´MANUFACTURER´ is subtype-of NOLOT called ´OWNER´;
```

This property is not a coincidence. It follows directly from the fact that the conceptual schema in these Binary Relationship approaches is perfectly capable of describing itself in the sense of chapter 2, sections 2.2 and 2.4, and chapter 3, section 3.5. We elaborate this further on.

(Note. A complete declaration of ´CAR-REGISTRATION´ appears in section E.6).

E.3.3. SEMANTICS.

Rule	Semantics	Description
(R2)	(S1)	NOLOT instances with the uniquely identifying names ´nolot-name` are added to the conceptual schema;
(R3)	(S2)	same as (S1) for LOT instances with the names ´lot-name`;
	(S3)	LOT and NOLOT are disjoint subtypes of the meta-object type ´OBJECT-TYPE` (!) - in other words, ´lot-name` must not already exist as a ´nolot-name`;
(R5)	(S4)	the two NOLOTs mentioned must exist already;
	(S5)	since subtyping is a transitive property, there must not be any closed "loop" in the subtype links;
(R6)	(S6)	same as (S4), mutatis mutandis;
	(S7)	same as (S5);
(R7)	(S8)	ROLE instances with ´role-name` are added; two ´role-name`s must be different when they are "on" the same NOLOT;
	(S9)	same as (S4);
	(S10)	the IDEA instance is then added with the uniquely identifying name ´idea-name`;
(R8)	(S11)	same as (S8);
	(S12)	the LOT and NOLOT mentioned must exist already;
	(S13)	same as (S10), mutatis mutandis;

	(S14)	the LOT mentioned must not appear in another BRIDGE declaration;
(R9)	(S15)	same as (S8), mutatis mutandis;
	(S16)	same as (S6);
	(S17)	same as (S10), mutatis mutandis;
(R10)	(S18)	a CONSTRAINT instance is added with the uniquely identifying name ´constraint-name`;
	(S19)	"sentence" must describe in a syntactically correct way a predicate (functor) which is to hold true at all times (for all states of the information base);
	(S20)	no two CONSTRAINTs may be in contradiction (this results in the impossibility to populate the information base);
	(S21)	the set of all CONSTRAINTs is considered to be an exhaustive list of all existing rules or constraints on all states of the information base, i.e. a binary relationship instance is entered in the information base if and only if no CONSTRAINT is violated;
	(S22)	the set of all constraints implies that all changes in the information base, which do not violate the declared CONSTRAINTs, are allowed;
(R20)	(S23)	The identifying-names must be unique within the conceptual schema, except the role-names, which have a more limited scope of uniqueness (cf. S8, S11, S15).

General Remarks on these semantics:

* People familiar with Knuth´s attribute mechanism [10] perhaps will recognize the similarity between the above way of describing semantical properties attached to syntactical tokens and Knuth´s attributes. See the literature [10] for more details.

* Apart from these "attribute" semantics there are semantics inherent to the primitive constructs and the way they are used in the model. Some are already more or less informally discussed in previous sections and will be only mentioned in passing.

* One of them is the concept of ´name`, such as ´nolot-name` or ´role-name`, for instance:

´name`: a term referring to an individual entity in the universe of discourse used for denoting a single entity. It is lexical in nature, that is there is a 1-1 correspondence with some string over a given alphabet. Nevertheless, it should be emphasized that the Binary Relationship approaches fully support homonyms as well as synonyms.

* At this point, a careful reader will have recognized that, strictly speaking, terminal grammar elements such as NOLOT or IDEA (commonly called "keywords") are also name instances (of primitive concepts), but on the meta-conceptual level. To avoid confusion in the syntax and resulting language these names do not appear in quotes.

* To make things complete, observe how the description of the semantics (S1 - S23) can themselves easily be viewed as constraints (both static and dynamic) in effect on this meta-conceptual level !

Important note:

As seen above, nothing prevents of course to view conceptual schemata themselves, or any other "abstract" language, as a universe of discourse in these approaches. It is possible this way to introduce several levels of description. However, at each level of description and in each state of the conceptual schema and information base, the conceptual schema and the part of the universe discourse formulation, it describes, are distinguishable by the environment. The reason is obvious, since the conceptual schema must be known to serve as the one and unique "contract" under which the information base may be approached and manipulated.

E.4. GRAPHIC FORMALISM.

Most investigators of the Binary Relationship approaches have adopted a graphic formalism to represent a part of the conceptual schema and information base, or more commonly, to represent a subset of the conceptual schema. As already explained in section E.1, a very clear distinction between non-lexical objects and lexical objects is made. This distinction is reflected in the graphic language. In this section we declare the graphic symbols used to represent the information structuring part of the conceptual schema in these approaches.

E.4.1. LINGUISTIC OBJECT TYPES.

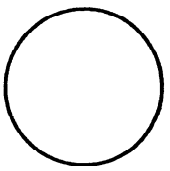


Figure E.3. Non-lexical object type symbol.

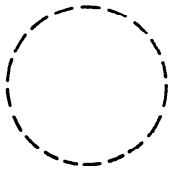


Figure E.4. Lexical object type symbol.

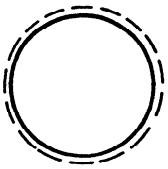


Figure E.5. Non-lexical object type symbol with a preferred ("natural") one-to-one bridge to a corresponding lexical object type. (This symbol is a "graphic macro construct" for the diagram construct shown in figure E.15).

E.4.2. BINARY RELATIONSHIP TYPES.

Figure E.6. Subtype link.

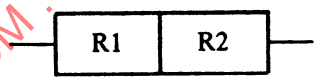


Figure E.7. Idea or bridge symbol; R1 and R2 are role names.

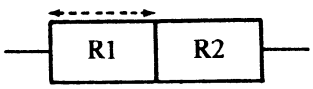


Figure E.8. Same, but R1 identifies R2.

E.4.3. CONSTRAINTS HAVING A DIAGRAMMATIC REPRESENTATION.

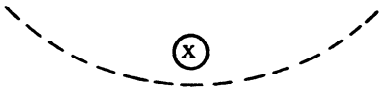


Figure E.9. Exclusion (between subtype links).

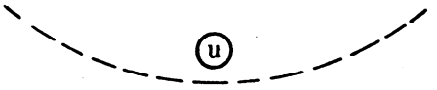


Figure E.10. Unique combination (of idea or bridge roles).

Figure E.11. Subset (between two role populations).

Figure E.12. Equality of population (of idea and bridge roles).

Of course, not all constraints will or can be pictorially represented in a (general) conceptual schema diagram. However, all are declared separately in some convenient language which refers to the linguistic object type names, binary relationship type names, etc, of the conceptual schema.

E.4.4. SOME EXAMPLES OF THE GRAPHIC FORMALISM SYMBOLS.

Turning to the diagram for the example of appendix B, let us illustrate the graphic formalism on a few simple excerpts from this diagram (cf. figure E.18).

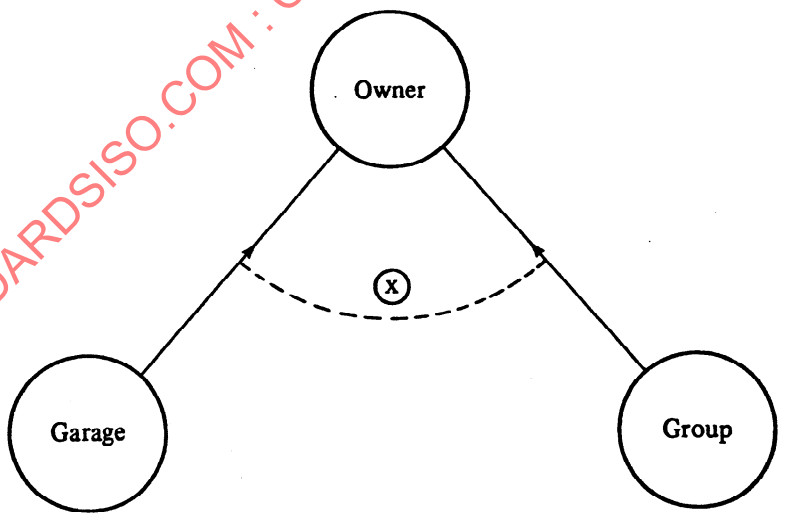


Figure E.13. A subtype; OWNER has, among others, a division between two exclusive sub-ownerships, GARAGE and GROUP.

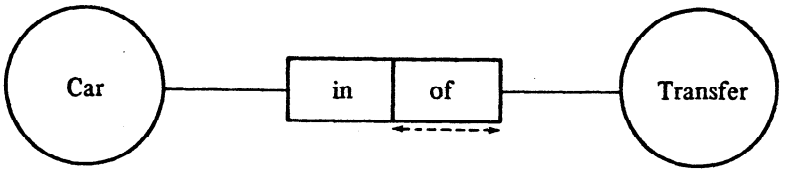


Figure E.14. An idea type; their occurrences are the real information carrying elements in these approaches.

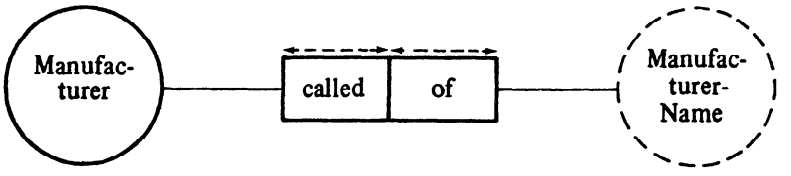


Figure E.15. Describes a 1-1 bridge: a manufacturer and his manufacturer-name identify each other.

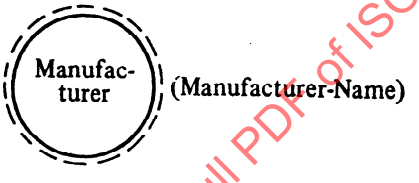


Figure E.16. This linguistic object type has an implicit, usually obvious and preferred unique name for each of its occurrences; it is the standard shorthand construct for constructs like the example in figure E.15.

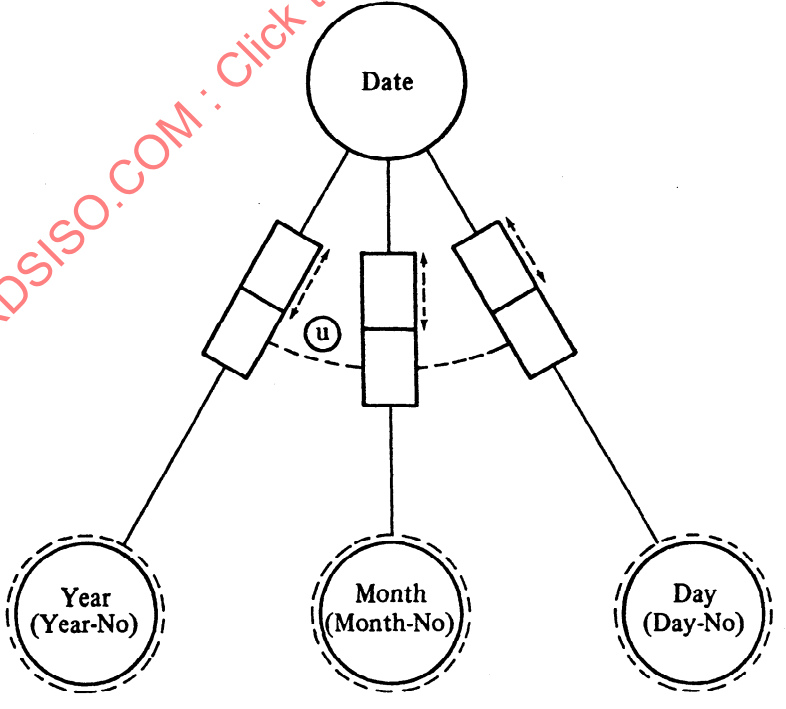


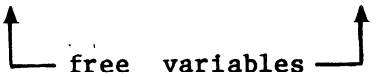
Figure E.17. Knowing a particular DATE (occurrence), we can uniquely determine its YEAR, MONTH or DAY, but also the combination of any YEAR, MONTH, DAY uniquely identifies any DATE.

E.5. MODELLING.

It will be clear there is no single "algorithm" to arrive at a conceptual schema. We limit ourselves here to some heuristics of general value and applicability.

One of the ways to design a conceptual schema in the Binary Relationship approaches is to take as starting point an informal perception of the universe of discourse. Select the necessary propositions about the universe of discourse and identify the free variables. Intuitively, these are terms (type names) referring to classes in the universe of discourse which may be replaced by particular (name-)instances in a sentence instance referring to entities that belong to the selected classes. For example:

"A car is made by a manufacturer"



Use the strings, representing the free variables, as possible names for types in the conceptual schema.

Identify the proposition sentences which talk about other proposition sentences. Some of these will result in additional linguistic object types and binary relationship types, while others result in constraints (sentences). For some there might be an option for either. The former are typically encountered where additional information is given about a particular sentence type (which can be "add"-ed), while the latter will restrict the population of existing sentence instances.

Thereafter, use the sentence occurrences expressing additional propositions about an entity world to check against instances derived from the conceptual schema. If one arrives at contradictions, then the conceptual schema will be modified to eliminate the contradictions (supposing, as always, that the propositions all hold in the universe of discourse).

Possibly extend the sentence occurrences referring to the universe of discourse. This extension is to be generated by the user together with the information system designer. This step is needed when the conceptual schema and information base is found to be an insufficiently complete view of the universe of discourse.

Repeat the above described steps until an acceptable degree of quality is reached.

E.6. EXAMPLE CONCEPTUAL SCHEMA.

E.6.1. GRAPHIC REPRESENTATION.

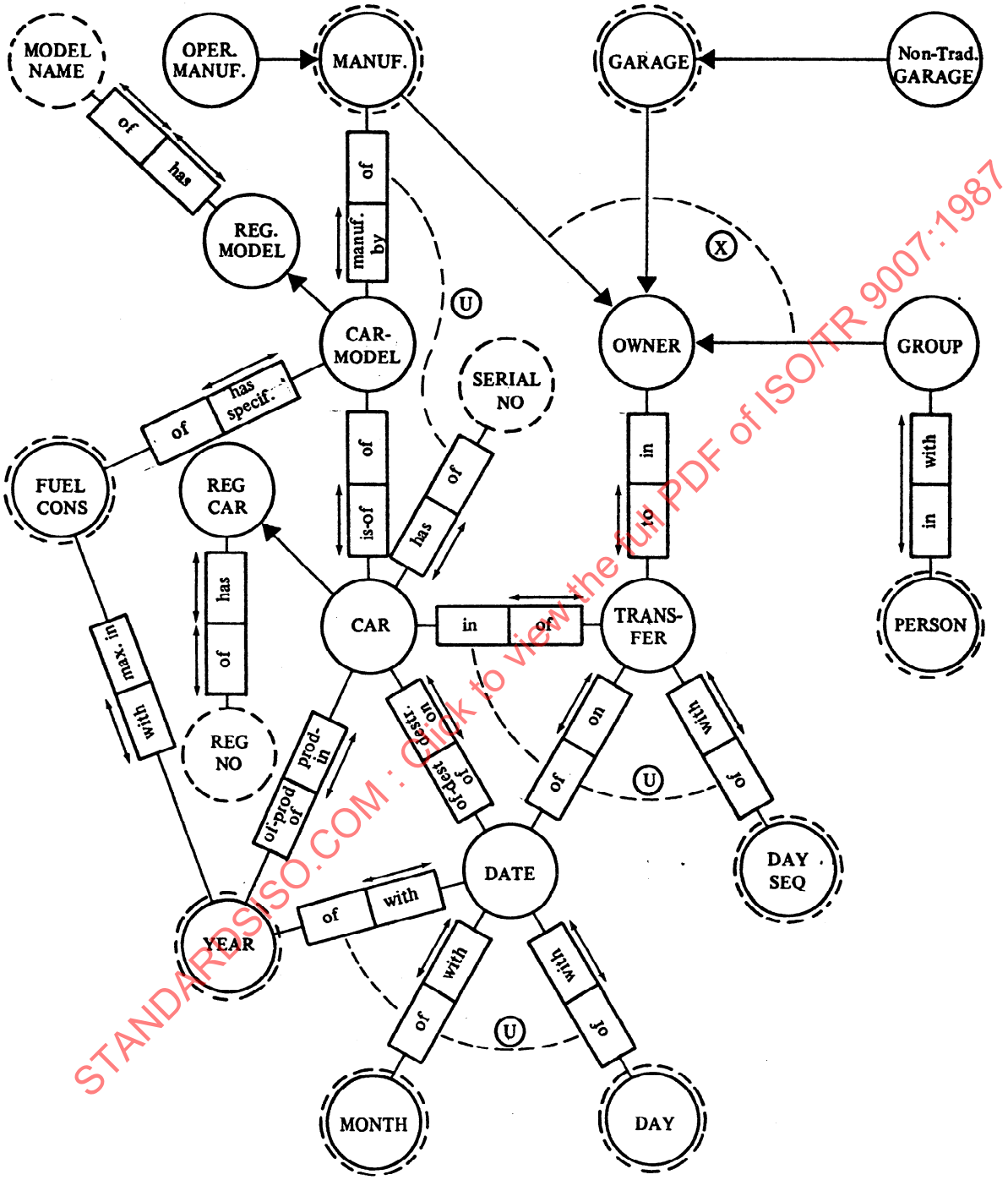


Figure E.18. Example conceptual schema.

E.6.2. LANGUAGE EXAMPLE.

A description of this schema (complete but for trivial repetitions) in the language of the grammar defined in section E.3 above, is as follows:

```
begin

add CONCEPTUAL-SCHEMA called 'CAR-REGISTRATION' ;

add NOLOT called {'MANUFACTURER' 'OPERATING-MANUFACTURER' 'REG-CAR'
                  'CAR' 'REG-MODEL' 'CAR-MODEL' 'FUEL-CONSUMPTION'
                  'DATE' 'YEAR' 'MONTH' 'DAY' 'TRANSFER' 'DAY-SEQ'
                  'OWNER' 'GARAGE' 'NON-TRADING-GARAGE' 'GROUP'
                  'PERSON'};

add LOT called   {'MANUFACTURER-NAME' 'REG-NO' 'SERIAL-NO' 'MODEL-NAME'
                  'FUEL-CONSUMPTION-AMOUNT' 'YEAR-NO' 'MONTH-NO'
                  'DAY-NO' 'SEQ-NO' 'GARAGE-NAME' 'PERSON-NAME'};

add NOLOT called 'OPERATING-MANUFACTURER'
        is subtype-of NOLOT called 'MANUFACTURER';

add NOLOT called {'MANUFACTURER' 'GARAGE' 'GROUP'}
        is subtype-of NOLOT called 'OWNER';

    Note: three other subtype declarations omitted here.

add IDEA (with-first ROLE (called 'manuf-by'
                           and on NOLOT called 'CAR-MODEL')
        and with-second ROLE (called 'of'
                           and on NOLOT called 'MANUFACTURER'))
        is called 'builds';

    Note: thirteen other idea declarations omitted here.

add BRIDGE (with-first ROLE (called 'called'
                             and on NOLOT called 'REG-CAR')
          and with-second ROLE (called 'of'
                             and on LOT called 'REG-NO'))
          is called 'registration';

    Note: two other explicit bridge declarations omitted here.

add BRIDGE (with-first ROLE (called 'called'
                             and on NOLOT called 'MANUFACTURER')
          and with-second ROLE (called 'of'
                             and on LOT called 'MANUFACTURER-NAME'))
          is called 'naming-of-model';

    Note: seven other implicit bridge declarations omitted here.

    Note: the list of constraints is given on the next pages

end.
```

The reader will have no difficulty supplying the omitted declarations from the diagram in Figure E.18. Note, that no phrases are present in this example.

In addition, the following list of constraint declaration bodies is conjectured to be complete and exhaustive with respect to the example of appendix B.

In the example in this section it is assumed that only the Registration Authority is permitted to manipulate the information system. This implies that no additional authorization rules are included in the example. Furthermore, the linguistic object type ´MANUFACTURER´ contains only manufacturers with permission. Thus we take ´having permission´ as the defining property of manufacturer, at least for the scope of interest of the Registration Authority. The linguistic object type ´CAR´ contains all cars known.

```
1. CONSTRAINT manuf-id
   is declared as
   begin
        MANUFACTURER-NAME of MANUFACTURER is unique;
        MANUFACTURER called MANUFACTURER-NAME is unique
   end;

2. CONSTRAINT number-of-operating-manufacturers
   is declared as
        number-of (OPERATING-MANUFACTURER) is not greater than 5;

3. CONSTRAINT non-op-manufacturer
   is declared as
   begin
        for each M : MANUFACTURER
          condition
            (M is not OPERATING-MANUFACTURER)
                                  excludes (M is current-owner-of CAR)
          holds
        end for
   end;

4. CONSTRAINT car-id
   is declared as
   begin
        REG-NO of REG-CAR is unique;
        REG-CAR called REG-NO is unique
   end;

5. CONSTRAINT eternal-reg-no
   is declared as
        delete of (REG-NO of, REG-CAR called)
        is not permitted;

6. CONSTRAINT mandatory-car-model
   is declared as
   begin
        CAR always is-of CAR-MODEL;
        CAR is-of CAR-MODEL is unique
   end;
```

```
7. CONSTRAINT mandatory-serial-no
   is declared as
   begin
        CAR always has SERIAL-NO;
        CAR has SERIAL-NO is unique
   end;

8. CONSTRAINT serial-no-manuf
   is declared as
        (SERIAL-NO of CAR and MANUFACTURER of CAR-MODEL of CAR)
                                                    is unique;

9. CONSTRAINT mandatory-car-owner
   is declared as
        CAR always in TRANSFER
        and
        OWNER always in TRANSFER;

10. CONSTRAINT mandatory-production-year
    is declared as
    begin
         CAR always produced-in YEAR;
         CAR produced-in YEAR is unique
    end;

11. CONSTRAINT registration-production-dates
    is declared as
    begin
         for each c : CAR
           condition
             YEAR-NO of reg-date-of (c) is in
             {YEAR-NO of-production-of c, (YEAR-NO of-production-of c) + 1}
           holds;
           if YEAR-NO of reg-date-of (c)
                                is equal to (YEAR-NO of-production-of c) + 1
           then condition
                  MONTH-NO of reg-date-of (c) is equal to '01'
                holds
           end if
         end for
    end;

12. CONSTRAINT single-destruction-date
    is declared as
         CAR destruction-on DATE is unique;

13  CONSTRAINT destruction
    is declared as
         YEAR-NO of DATE of-destruction-of CAR
                                is not less than (YEAR-NO of today) - 2;

14. CONSTRAINT car-model-id
    is declared as
    begin
         MODEL-NAME of REG-MODEL is unique;
         REG-MODEL called MODEL-NAME is unique
    end;
```

```
15. CONSTRAINT eternal-model-name
    is declared as
         delete of (MODEL-NAME of, REG-MODEL called)
         is not permitted;

16. CONSTRAINT mandatory-car-model-manufacturer
    is declared as
    begin
         CAR-MODEL always manuf-by MANUFACTURER;
         CAR-MODEL manuf-by MANUFACTURER is unique
    end;

17. CONSTRAINT no-other-model-manuf
    is declared as
         delete of (CAR-MODEL manuf-by, MANUFACTURER of)
         is not permitted;

18. CONSTRAINT must-know-fuel-cons
    is declared as
    begin
         CAR-MODEL always has-specified FUEL-CONSUMPTION;
         CAR-MODEL has-specified FUEL-CONSUMPTION is unique
    end;

19. CONSTRAINT fuel-consumption-id
    is declared as
    begin
         FUEL-CONSUMPTION-AMOUNT of FUEL-CONSUMPTION is unique;
         FUEL-CONSUMPTION has FUEL-CONSUMPTION-AMOUNT is unique
    end;

20. CONSTRAINT min-max-fuel-cons
    is declared as
         FUEL-CONSUMPTION-AMOUNT is
                              (not greater than 25 and not less than 4);

21. CONSTRAINT average-fuel-consumption
    is declared as
    begin
         if MONTH-NO of today is equal to '01'
         then for each M : MANUFACTURER
           condition
             average (FUEL-CONSUMPTION of MODEL of CAR
                          (with CAR-MODEL maunf-by M and produced-in
                                                ((YEAR-NO of today) - 1)))
                  is not greater than
                       (FUEL-CONSUMPTION max-in (YEAR-NO of today) - 1)
           holds;
           on violation do
             send-message-to (M)
           end for
         end if
    end;
```

Note: Average is standard function which causes the parameter expression to be evaluated as a multiset (rather than a set). A multiset is a mathemathical concept that describes a set in which the same element may occur more than once.

```
22. CONSTRAINT fuel-consumption-year
    is declared as
         YEAR with FUEL-CONSUMPTION is unique;

23. CONSTRAINT garage-id
    is declared as
    begin
         GARAGE-NAME of GARAGE is unique;
         GARAGE called GARAGE-NAME is unique
    end;

24. CONSTRAINT garage-suppliers
    is declared as
    begin
         for each G : GARAGE
           cars-of-Gar := current-stock-of (G);
           condition
             number-of (MANUFACTURER of CAR-MODEL of cars-of-garage)
                                                  is not greater than 3
           holds
         end for
    end;

25. CONSTRAINT garage-not-trading
    is declared as
    begin
         for each G : GARAGE
           condition
            (G is NON-TRADING-GARAGE) implies
                                    current-stock-of (G) is empty
           holds
         end for
    end;

26. CONSTRAINT mandatory-person-in-group
    is declared as
         PERSON always in GROUP;

27. CONSTRAINT person-id
    is declared as
    begin
           PERSON-NAME of PERSON is unique;
           PERSON called PERSON-NAME is unique
    end;

28. CONSTRAINT owner-id
    is declared as
         TRANSFER to OWNER is unique;
```

```
29. CONSTRAINT owner-subtypes
    is declared as
         OWNER is equal to (MANUFACTURER union GARAGE union GROUP);

30. CONSTRAINT owner-exclusive-subtypes
    is declared as
         (GROUP intersection GARAGE is empty)
         and
         (GROUP intersection MANUFACTURER is empty)
         and
         (GARAGE intersection MANUFACTURER is empty);

31. CONSTRAINT transfer-id;
    is declared as
         (CAR in TRANSFER and DATE of TRANSFER and SEQ-NO of TRANSFER)
                                                        is unique;

32. CONSTRAINT transfer-single-car
    is declared as
    begin
         TRANSFER always of CAR;
         TRANSFER of CAR is unique
    end;

33. CONSTRAINT transfer-single-date
    is declared as
    begin
         TRANSFER always on DATE;
         TRANSFER on DATE is unique
    end;

34. CONSTRAINT transfer-single-day-seq
    is declared as
    begin
         TRANSFER always with DAY-SEQ;
         TRANSFER with DAY-SEQ is unique
    end;

35. CONSTRAINT date-id
    is declared as
         (YEAR of DATE and MONTH of DATE and DAY of DATE) is unique;

36. CONSTRAINT single-date-descr
    is declared as
    begin
         DATE with YEAR is unique;
         DATE with MONTH is unique;
         DATE with DAY is unique
    end;

37. CONSTRAINT year-id
    is declared as
    begin
         YEAR-NO of YEAR is unique;
         YEAR called YEAR-NO is unique
    end;
```

```
38. CONSTRAINT month-id
    is declared as
    begin
         MONTH-NO of MONTH is unique;
         MONTH called MONTH-NO is unique
    end;

39. CONSTRAINT day-id
    is declared as
    begin
         DAY-NO of DAY is unique;
         DAY called DAY-NO is unique
    end;

40. CONSTRAINT day-seq-id
    is declared as
    begin
         SEQ-NO of DAY-SEQ is unique;
         DAY-SEQ called SEQ-NO is unique
    end;

41. CONSTRAINT no-transfer-after-destruction
    is declared as
    begin
         for each c : CAR do
           condition
             YEAR-NO.MONTH-NO.DAY-NO of DATE of TRANSFER of c is less than
                      YEAR-NO.MONTH-NO.DAY-NO of DATE of destruction-of c
           holds
         end for
    end;

42. CONSTRAINT possible-transfers
    is declared as
    begin
         for each T : TRANSFER
           P := previous-owner-in (T);
           condition
             (T is TRANSFER to MANUFACTURER implies P is empty)
             and
             (T is TRANSFER to GARAGE implies P is not GARAGE)
             and
             (T is TRANSFER to GROUP implies P is not MANUFACTURER)
           holds
         end for
    end;
```

A few notes on the language example:

1. Current-owner-of, current-stock-of, cars-of-garage, and previous-owner-in are functions that can be defined in the same language. They are used here as "macros". Note, that current-stock-of returns a set of cars which does not include destroyed cars.

2. The super-types REG-MODEL and REG-CAR, with their associated

bridges and the "no delete" constraints 5 and 15 guarantee the uniquenees of MODEL-NAME and REG-NO for all times.

3. The (deducible) registration-date is equal to the date of the first transfer of a car.

4. The identification of TRANSFER also acts as identification of OWNER and of GROUP, if a GROUP is involved.

5. The keyword "today" is a standard predefined non-lexical object of type DATE which always contains the current (system) date.

6. There is supposed to exist a collating sequence (natural order) for the occurrences of the LOT ´MONTH-NO´. Furthermore YEAR-NO and DAY-NO are supposed to be populated with natural numbers. This needs to be done in such a way that the concatenation YEAR-NO.MONTH-NO.DAY-NO allows (a linear order) comparison between dates so described.

E.7. CHECK LIST FOR THE CONCEPTUAL SCHEMA.

The following analysis illustrates whether or not the necessary propositions about the universe of discourse are captured in the conceptual schema. A D implies that the assertion is described in the construct descriptions in the conceptual schema. A numbered C refers to a defined constraint. An * refers to remarks at the end of this section.

CHECK		NECESSARY PROPOSITIONS
D	1.	The universe of discourse to be described has to do with the registration of cars and is limited to the scope of interest of the Registration Authority.
C1	2.	Each car manufacturer has a unique name.
*	3.	New car manufacturers can start operation provided they have the permission of the Registration Authority.
*	4.	The Registration Authority cannot withdraw the permission.
C2	5.	At any time not more than five autonomous manufacturers may operate.
C3	6.	Manufacturers may cease to operate, provided they do not own cars anymore.
D	7.	Each car manufacturer constructs cars in several models.
C6	8.	A car is of a particular model.
*	9.	A manufacturer gives a serial number to each car he produces.
C8	10.	This serial number is unique for all cars of one manufacturer.
*	11.	A newly produced car is registered by the Registration Authority as soon as practicable.
C9/ C29	12.	At this time the car is registered as belonging to the manufacturer which produced it. Therefore the first owner will be the manufacturer who produced the car.
*	13.	Only the Registration Authority will assign a registration number to each registered car.
C4/5	14.	This registration number is unique for all cars for all time.
C10	15.	A car has a year of production.
C11	16.	Only in January may a car be registered as being produced in the previous year.
C12	17.	Cars can be destroyed whereupon the date of destruction is recorded.

C13	18.	The car's history has to be kept until the end of the second subsequent calendar year after its destruction. Thereafter, it is removed from the registered information.
C14/ C15	19.	The name of the car model is unique for all car models for all time.
C16/ C17	20.	Any specific car model is constructed by only one manufacturer.
*	21.	From time to time new models will be introduced.
C18	22.	All cars of the same car model have the same fuel consumption.
C19	23.	This fuel consumption must be known to the Registration Authority.
C20	24.	The fuel consumption of a car will be between 4 and 25 litres per 100 km.
C21/ C22	25.	The fuel consumption averaged over all individual cars produced by a particular manufacturer in a particular year is required not to exceed a maximum value which is the same for each manufacturer.
C22	26.	The maximum fuel consumption rate may change from year to year.
C21	27.	At the end of January a message is sent to a manufacturer who has failed to meet this requirement in the previous year.
C23	28.	Each garage has a unique name.
*	29.	New garages may be established.
C29	30.	Garages may own cars.
C24	31.	A garage must not have, at any time, cars registered as belonging to the garage, from more than three manufacturers (which three does not matter, and for a particular garage may vary with time).
C25	32.	An existing garage may be closed down, provided it does not have any cars registered to it.
C29	33.	A particular person may have one or more cars registered as belonging to him or her.
C26	34.	It is also possible for two or more people to have one or several cars registered as belonging to them jointly and simultaneously.
C27	35.	People have unique names.

C9/ C26 36. People are only known to the Registration Authority if they own or have owned (one or more) cars, which still are known to the Registration Authority.

C29/ C30 37. At any time a car is owned by either

- its manufacturer,
- a garage,
- a person,
- a group of persons,

but not jointly by two or more of these categories.

C9/ C33 38. Transfer of ownership is registered including the date of transfer, the previous owner(s) and the new owner(s).

C41 39. Transfer of ownership cannot take place anymore after a car's destruction.

* 40. However, transfer of ownership may be recorded after the car's destruction, provided the transfer of ownership took place before the car's destruction.

C42 41. Each manufacturer distributes new cars to several independent garages, each which may receive cars from more than one manufacturer.

C42 42. Therefore a garage always will be a car's second owner.

C42 43. Manufacturers do not distribute cars to other manufacturers or directly to people.

C42 44. Each garage may sell - i.e., cause transfer of registered ownership of - new or used cars to people, and may buy - i.e., cause transfer of registrered ownership of - cars from people.

C42 45. Garages are not allowed to sell cars to other garages.

C42 46. Garages are not allowed to sell cars to manufacturers.

C42 47. People can sell cars to other people or buy cars from other people.

Remarks for the Binary Relationship approaches:

* No authorization rules are included. For example, checks 3, 4, 9, 11, and 13.

* Prescriptive rules for interactions are not part of the conceptual schema. For example, checks 21, 29, and 40.

E.8. REFERENCES.

[1] KENT, W. ´Data and Reality`, North-Holland Publishing Company, 1978.

[2] BUBENKO, J.A. ´The temporal dimension in information modelling` In: Architecture and models in database management systems, Proceedings IFIP TC2 Conference, Nice, 1977; North-Holland Publishing Company.

[3] ABRIAL, J.R. ´Data Semantics`, In: Data Base Management, Proceedings IFIP TC2 conference, Cargese, 1974; North-Holland Publishing Company.

[4] BRACCHI, G., PAOLINI, P. and PELAGATTI, G. ´Binary Logical Associations in Data Modelling` In: Modelling in data base management systems, proceedings IFIP TC2 Conference, Freudenstadt 1976; North-Holland Publishing Company.

[5] SENKO, M.E. ´Conceptual schema, abstract data structures, enterprise descriptions`, In: International computing symposium 1977; North-Holland Publishing Company.

[6] DURCHHOLZ, R. and RICHTER, G. ´Concepts for Data Base Management Systems` In: Data Base Management, Proceedings IFIP TC2 conference, Cargese, 1974; North-Holland Publishing Company.

[7] FALKENBERG, E., BREUTMAN, B. and MAUER, R. ´CSL: a language for defining conceptual schemas`, In: Data Base Architecture, Proceedings IFIP TC2, Venice 1979, North-Holland Publishing Company.

[8] NIJSSEN, G.M. ´A framework for advanced mass storage applications`, In: Medinfo 80, Proceedings of the Third World Conference on Medical Informatics, Tokyo 1980, North-Holland Publishing Company, 1980.

[9] MEERSMAN, R. and VERMEIR, D. ´RIDL Reference Manual`, In: Data Management Research Report, January 1980, Control Data Belgium.

[10] KNUTH, D.E. ´Semantics of Context-free Languages`, In: Math. Systems Theory J. 2:2, pp. 127 - 146.

APPENDIX F. THE INTERPRETED PREDICATE LOGIC APPROACHES.

F.1. EMPHASIS OF THE APPROACHES.

The Interpreted Predicate Logic (IPL) approaches to conceptual schema description are based on the fundamental principle that not only the conceptual schema but the information base as well and, thus, the entire conceptual schema and information base is a collection of abstract things called "sentences" which are in practice represented by strings of characters that are themselves encoded in a suitable set of physically real states in the storage media of an information system.

The guiding principle in the design of these approaches is the use of an interpreted, axiomatized, deductive, formal system of logic that places minimal demands, in principle, on the information processor external to the conceptual schema and information base itself. In practice, of course, many theoretically unnecessary constructs may well be built into an actual information processor for reasons of efficiency and costs.

The principle enunciated in the preceding paragraph can be satisfied in a variety of ways and a second guiding principle is necessary to permit choice amongst these various possibilities. The relevant principle derives from the requirement that conceptual schemata formulated according to these approaches be easy to use and to understand by a variety of users. Thus, a mechanism for adding constructs of arbitrary complexity to the formal system is necessary so that users of the system may interact with it at any desired level of aggregation of concepts [1, 2, 3]. In order for this principle to be observed without violating the first principle, the formal system must have within itself a mechanism for incorporating the definitions of new constructs in terms of those already present.

For the purpose of this exposition the information system in use is a combination of the printed pages in this Report and the mind of the reader. The characters used to express sentences are here chosen from IS 646 ("7-bit input/output coded character set for information interchange") in order to conform to existing ISO practices, although it will be evident to the reader that a larger,more comprehensive character set would enhance readibility.

Considering the objectives for a conceptual schema, cited in section 1.9 of chapter 1, the following remarks could be made about the IPL approaches:

1. "To provide a common basis for understanding the general behaviour of the universe of discourse."
   A formal language of logic, as demonstrated in this chapter, is sufficient to completely describe the universe of discourse including all classes, rules, and constraints of its behaviour.

2. "To define the allowed evolution and manipulation of the information about the universe of discourse."
   The use of the concepts of a formal system of logic, as outlined in the IPL approaches, makes it possible to completely describe the dynamic rules and constraints of the conceptual

schema and information base, as well as the static rules and constraints, and thus gives a complete control over the manipulation of the conceptual schema and information base.
As, fundamentally, there is no difference in the mechanisms needed for insertion, deletion, etc, of sentences, whether they are thought to be in the conceptual schema or in the information base, there will be no extra difficulties or complexities in changing the conceptual schema to reflect changes in the abstraction system.

3. "To provide a basis for interpretation of external and internal syntactical forms which represent the information about the universe of discourse.
To serve this role, it is needed that a conceptual schema contains the deepest structure of the problem description and formulates all rules and constraints applicable in the conceptual schema and information base. This requirement is fully met by the IPL approaches.
Although a formal language of logic, as demonstrated in this chapter, is sufficient to completely describe the universe of discourse, such a language may be extended with, or (partly) replaced by, the more complex constructs mentionned above to ease users of the system (e.g.[4]). Often a graphic formalism covering major aspects of the conceptual schema is added to ease communication with users in discussing aspects of the conceptual schema and information base [1, 2, 3].

4. "To provide a basis of mappings between and among external and internal schemata."
The maximum degree of stability for the purpose meant here is reached with semantically irreducible constructs and constructs free of additional constraints. This requirement is fully met by the IPL approaches.

F.2. PRIMITIVE CONCEPTS OF THE APPROACHES.

In the IPL approaches to conceptual schema and information base, there are two fundamental concepts for the universe of discourse, entities and propositions, as defined in chapter 2, section 2.1. As far as the universe of discourse to be modelled is concerned, the only things that exist are entities, and the only situations that can be affirmed or denied about entities in the universe of discourse are propositions.

The establishment of a formal system of logic to describe the universe of discourse incorporates the requirements of chapter 3, section 3.2. As stated there, the axioms and deduction rules are chosen in such a way as to result in each axiom being interpreted as a true assertion about the universe of discourse and each sentence immediately deducible from a set of sentences interpreted as true assertions about the universe of discourse is itself interpreted as a true assertion about the universe of discourse. The only deduction rule required is that given in section F.3.3 and it does preserve truth. The axioms, however, are to a large extent chosen by the designer of the conceptual schema for a particular universe of discourse and it is incumbent on that designer to insure that the interpretation is correct.

The outline of the formal language used is prescribed only to the extent necessary to establish the fundamental categories of grammatical constructs, and to insure interpretability by a universal Turing machine. No concrete syntax is exposed (except as necessary to detail the example), so that any formal language adhering to the abstract syntax discussed can be used to carry out the program described.

What is presented in this chapter follows from a long line of development of formal logic in the tradition of Boole (1847) [5], Jevons (1864) [6], Frege (1879) [7], Peirce (1885) [8], Peano (1908) [9], Zermelo (1908) [10], and Whitehead and Russell (1910-1913) [11]. The contemporary exposition closest to the spirit of these approaches is that of Quine (1951) [12]. Its metaphysical presuppositions are those of the modern logistic school, closely related to the modern counterpart of naive realism. Plato, where are you when we need you?

F.3. GRAMMAR AND SEMANTICS.

The fundamental notions of an abstract syntax for a formal language have already been presented in chapter 3, section 3.3. Those notions are amplified in section F.3.1 below in a form suited to the IPL approaches.

To attribute meaning (semantics) to various expressions in the language, it is necessary to start with a (hopefully small) set of undefined concepts known as primitives. Other concepts then have meanings which are derivable from the informally understood primitive concepts through the introduction of formal definitions. The relevant essentials of the meaning of each primitive concept is formally captured through the assertion of axioms assumed to be valid. This is exactly the situation that obtains in elementary synthetic geometry where the concepts of points, lines and planes are taken as primitive (undefined) concepts, more complicated constructs such as triangles and circles are defined in terms of these primitives and axioms (Euclid's postulates) are stated to assert the properties of points, lines and planes.

F.3.1. ABSTRACT SYNTAX.

As defined in section 2.1 of chapter 2 and section 3.3 of chapter 3 (repeated here for the convenience of the reader), certain general concepts are primitive notions that characterize the abstract syntax (grammar) of any language. They are:

TERM
A linguistic object that refers to an entity;

SENTENCE
A linguistic object which expresses a proposition;

FUNCTOR
A linguistic object that refers to a function on other linguistic objects taking as arguments (input) a list of linguistic objects (terms, sentences, functors) and yielding as a value (output) a single, uniquely determined linguistic object (term, sentence, functor).

It is the functor that permits the construction of linguistic objects of arbit-

rary complexity. All that appears to be necessary for a conceptual schema and information base are first level functors, those functors where the arguments are all selected from among the terms and sentences (never functors) and the value is always a term or a sentence (never a functor). However, nothing in what is explicated below will prohibit the introduction of higher level functors in the event such constructs turn out to be useful for a conceptual schema and information base.

Functors are pure if their argument lists are always either all sentences or all terms, not a mixture. In addition, functors are substitution functors if there exists a pattern into which the elements of the argument list can be substituted. Examples are: "If ... Then ---" and "... + ---". Finally, functors may be fixed or not depending on whether the argument list is fixed in length or not.

Elementary functors are fixed, first level, substitution functors. There are four kinds of pure elementary functors. They are:

Arguments	Value	Name	Examples
Terms	Term	Operator	... + --- ...'s mother
Terms	Sentence	Predicate	... < --- ... is red
Sentences	Sentence	Connective	... and --- not ...
Sentences	Term	Subnector	"..." the probability that ...

It should be apparent that each of the above described pure elementary functors has application to conceptual schemata. Two other kinds of functors that are elementary but not pure have important application to conceptual schemata: quantifiers and abstractors. Noting that variables are a kind of term (cf. chapter 3, section 3.3):

Arguments	Value	Name	Examples
Variable + sentence	Sentence	Quantifier	For all ... ---
Variable + sentence	Term	Abstractor	The ... such that ---

Variables are the analogues of natural language pronouns and refer to unspecified, indeterminate entities in the universe of discourse, exactly in the same fashion that pronouns in natural language refer to unspecified, indeterminate things in sentences with anaphora (as in "It is red").

Just as in natural language one deals with dangling pronouns by providing some kind of referent, the same is done in formal languages. Quantifiers and abstractors fill that role as the examples indicate. One can repair the anaphora shown above by asserting (falsely but now meaningfully) "Whatever it is, it is red" Paraphrasing, one has "For every it, it is red" It is a trivial step to "For all ... ... is red."

Sentences and terms, however constructed, may be either open or closed depend-

ing on whether they have free variables or not, a free variable being the analogue of a dangling pronoun. Thus, "..." is free in "... is red" but not in "For all ... ... is red" Only closed sentences can give unambiguous interpretation and for that reason will be the only kind of sentences appearing in a conceptual schema and information base. It goes beyond the scope of this chapter to provide a detailed descripton of open and closed sentences and how variables are bound (see Quine [12] for details). It is sufficient to note that any sentence can have all its free variables eliminated by prefixing sufficient quantifiers.

For a purely extensional system that avoids modalities (e.g., "it is necessary that", "it is known that", "it is obligatory that", etc.), the only concepts required as primitives are variables (a kind of term, as noted), the connectives "Not..." and "If ... Then ---", the universal quantifier "For all ... ---", the abstractor "The ... such that ---", and a suitable supply of appropriate predicates. Other choices for primitives are quite possible and nothing fundamental in what follows precludes alteration of the primitives. For example, proper names can be introduced as primitive terms, but to simplify the discussion here they will be avoided. It has been shown by Russell [13] that proper names can always be eliminated in favor of predicates through the particular abstractor known as a "descriptor" (the exemplar abstractor above), and defining the proper name itself in context.

The essentials of this abstract syntax are as formulated by Curry and Feys [14] and the explicit details are derived from Anderson and Belnap [15]. It should be apparent that considerable detail is required to establish a particular concrete syntax.

F.3.2. CONCRETE SYNTAX.

In contrast to the other examples in this report, the concrete syntax for any version of the IPL approaches cannot be fully explicated in the syntax notation, presented in appendix C, as essential aspects of the language are not context free. What follows in this section is a sketch of a concrete syntax sufficient for understanding of the example exposed in section F.6. Some of it is necessarily discursive here in view of the context sensitivity, but it must be understood that the complete syntax can be made fully formal and can be expressed fully in any language that is suitable for expressing conceptual schemata using an IPL approach.

The language described below will be given the unimaginative name of "L". It is not likely to exhibit the precise form that will be useful in practice as the choice of syntactical detail here has been made on the basis of a desire to limit the character set to emphasize the exposition.

Thus:

```
upper-case-letter  = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
                     "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
                     "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z".

lower-case-letter  = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
                     "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
                     "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z".
```

```
digit                = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".

prime                = "´´".          Note that this means a single prime
                                      character ´.

point                = ".".

other-mark           = "&" | "(" | ")" | "*" | "+" | "-" | "/" | ":" | ";" |
                       "," | "<" | "=" | ">" | "[" | "\" | "]" | "{" | "}".
```

In the context of an IS 646 representation, one may use the control character "0001000" followed by the graphic character "1111101", yielding "backspace, underline", and so form:

```
upper-case-italic    = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
                       "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
                       "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z".

lower-case-italic    = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
                       "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
                       "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z".
```

Collecting, one has:

```
mark                 = upper-case-letter | lower-case-letter | digit | prime |
                       point | upper-case-italic | lower-case-italic | other mark.
```

A number of possible characters from the IS 646 graphics are not marks of L as defined here because they are unnecessary in the example. The blank character, "0000010", however, is never one of the marks of L. In this respect L is like FORTRAN where blanks may appear anywhere throughout the text to provide readability but have no significance. Of course, even this is not an essential property of L. Nothing in any concrete syntax is fundamental.

Proceeding:

```
string               = mark {mark}.

variable             = (upper-case-italic {prime}) | (lower-case-italic {prime}).

compound-symbol      = upper-case-letter {lower-case-letter} {prime}.

number-symbol        = {digit} [point] {digit}.

                           Note, provided the entire string is neither preceded
                           nor followed by a digit or a point.

symbol               = variable | compound-symbol | number-symbol | other-mark.

expression           = symbol {symbol}.
```

A little reflection will convince the reader that any string that is an expression can be decomposed into symbols in exactly one way.

```
primitive-predicate  = "Pr" {lower-case-letter} {prime}.
```

```
prime-variable-match = (prime variable) | (prime prime-variable-match variable).

atomic-sentence      = "Pr" {lower-case-letter} prime-variable-match.

primitive-sentence   = atomic-sentence |
                       ("For all" variable primitive-sentence) |
                       ("Not" primitive-sentence) |
                       ("If" primitive-sentence "Then" primitive-sentence).
```

An occurrence of a variable in a primitive-sentence is exactly what it informally sounds like but is a fundamentally context sensitive aspect of L. There is no occurrence of "x" in "x'". With that caveat, an occurrence of a variable is bound in a primitive-sentence if there is some primitive-sentence containing the occurrence of the variable that is part or all of the primitive-sentence in question and begins with the expression "For all" followed by the variable. Then, an occurrence of a variable in a primitive-sentence is free in the primitive-sentence if it is not bound in the primitive-sentence. A variable is a free variable of a primitive-sentence if it has a free occurrence in the primitive-sentence.

A primitive-sentence is open if it has at least one free variable, otherwise, it is closed.

Everything that can be asserted in L can be asserted by closed primitive-sentences. However, even the simplest assertions become lengthy, cumbersome and impossible to read when expressed as primitive-sentences. Therefore, definitions are introduced:

definition — Any expression is a definition if it is asserted to be so by an authorized source.

Excepting only a small number of special cases necessary to the initial bootstrap, cases that need not obtrude on this exposition, all definitions will be closed sentences in one of the following two forms:

expression "Iff" expression.

expression "=" expression.

As will be seen later, the symbol "Iff" will be interpreted as the connective of the biconditional, and its incorporation between two sentences asserts the logical equivalence of the sentences. A sequence of definitions in this first form is constructed so that the right hand expression (definiens) of each definition in the sequence is either a primitive-sentence or the left hand expression (definiendum) of some earlier definition in the sequence. In view of the intersubstitutability of logically equivalent expressions, very complex assertions can be made in compact form and shown to be logically equivalent to a primitive-sentence.

A quite similar process occurs for definitions of the second form as the symbol "=" will be interpreted as the predicate of identity and the same kind of intersubstitutability applies. Of course in this situation the expressions on both sides of the identity sign are terms rather than sentences. Using the device of contextual definition (Russell [13]), terms can be defined in every place that a variable (also a term) can appear in an atomic-sentence and so anything that can be asserted about the entity denoted by the term can be interpreted.

Given a sequence of definitions (the subject of section F.5.2), there is little more to say about the concrete syntax of L. For the sake of completeness the following concepts, given import in section F.5, are useful to introduce here in order to demonstrate that they are, in fact, syntactical concepts. Thus:

axiom	Any closed sentence is an axiom if it is asserted to be so by an authorized source (as in chapter 3, section 3.3). All definitions are axioms;
proof	Any sequence of closed sentences is a proof if for each sentence Q, either Q is an axiom or there are sentences P and "If P Then Q" prior to Q in the sequence;
theorem	Any sentence for which there exists a proof containing that sentence is a theorem.

While there is, in general, no algorithm for finding a proof that contains a given sentence and therefore no decision procedure for determining whether a candidate sentence is a theorem or not, the definition of "theorem" is syntactic and there is a mechanism for determining whether a given sequence is a proof and contains a candidate. Having defined "theorem", the discussion must now turn to "truth", a different matter entirely.

F.3.3. SEMANTICS.

Assuming an explicit concrete syntax yielding a formal language, L, the semantics of L is established by assigning meaning to its primitive constructs, as outlined in chapter 3, section 3.4. From that assignment all other semantic notions are derived from interpretation of the formal definitions of the derived constructs.

The interpretation of variables, the only primitive terms, has already been given (cf. chapter 3). The interpretation of the two primitive connectives and the primitive quantifier are obvious and are the usual ones intended by the conventional first order logic with the caution that the conditional "If ... Then ---" is the strict truth-functional conditional which asserts a true proposition so long as it is not the case that "..." asserts a true proposition while "---" asserts a false proposition.

The deduction rules must preserve truth. The latter is simple; use the rule of detachment:

from "..." and "If ... Then ---" deduce "---".

F.4. GRAPHIC FORMALISM.

Some IPL approaches (e.g. [1, 2, 3]) have adopted a graphic formalism to represent parts of the universe of discourse or its description. In most cases the graphic formalism is used to show the most important propositions - those that are referred to by the atomic-sentences - and the "entity pattern" they constitute.

The graphic formalism demonstrated in this chapter is taken from [3]. Here an entity is shown as a circle; the proposition is shown as a line connected to the entity or entities. The same formalism is adopted to show patterns of propositions about entities. For example the proposition about two entities as described by

Pra´´xy

is shown in figure F.1:

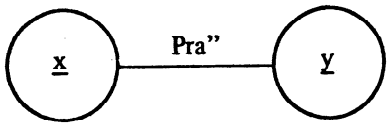


Figure F.1. Graphic representation of a proposition about two entities.

A proposition about three entities as described by

Prb´´´xyz

is shown in figure F.2:

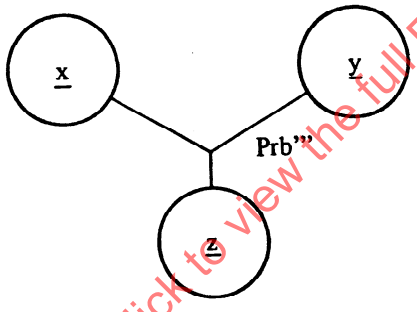


Figure F.2. Graphic representation of a proposition about three entities.

F.5. MODELLING.

F.5.1. CLASSIFICATION OF AXIOMS.

From among the sentences of L certain designated closed sentences are selected as axioms. One particular set, A1, constitutes the axioms of logic and mathematics. Application of the rules of deduction produces an additional set of sentences that are the theorems deducible solely on the basis of the axioms A1. While a wide variety of different sets of sentences can be chosen to serve as A1, the totality of theorems will be the same in each case as a general rule. It is a reasonable assumption that every conceptual schema and information base will include the same A1.

It should be noted that most of the "axioms" of A1 are actually axiom schemata, metalanguage assertions that every sentence of such and such a form is an axiom. Thus, while there are a finite number of such schemata, there are actually an infinite number of axioms in A1. For example, every sentence of the form:

If (If Not ... Then ...) Then ...,

where "..." is replaced throughout by an arbitrary sentence, is an axiom of A1.

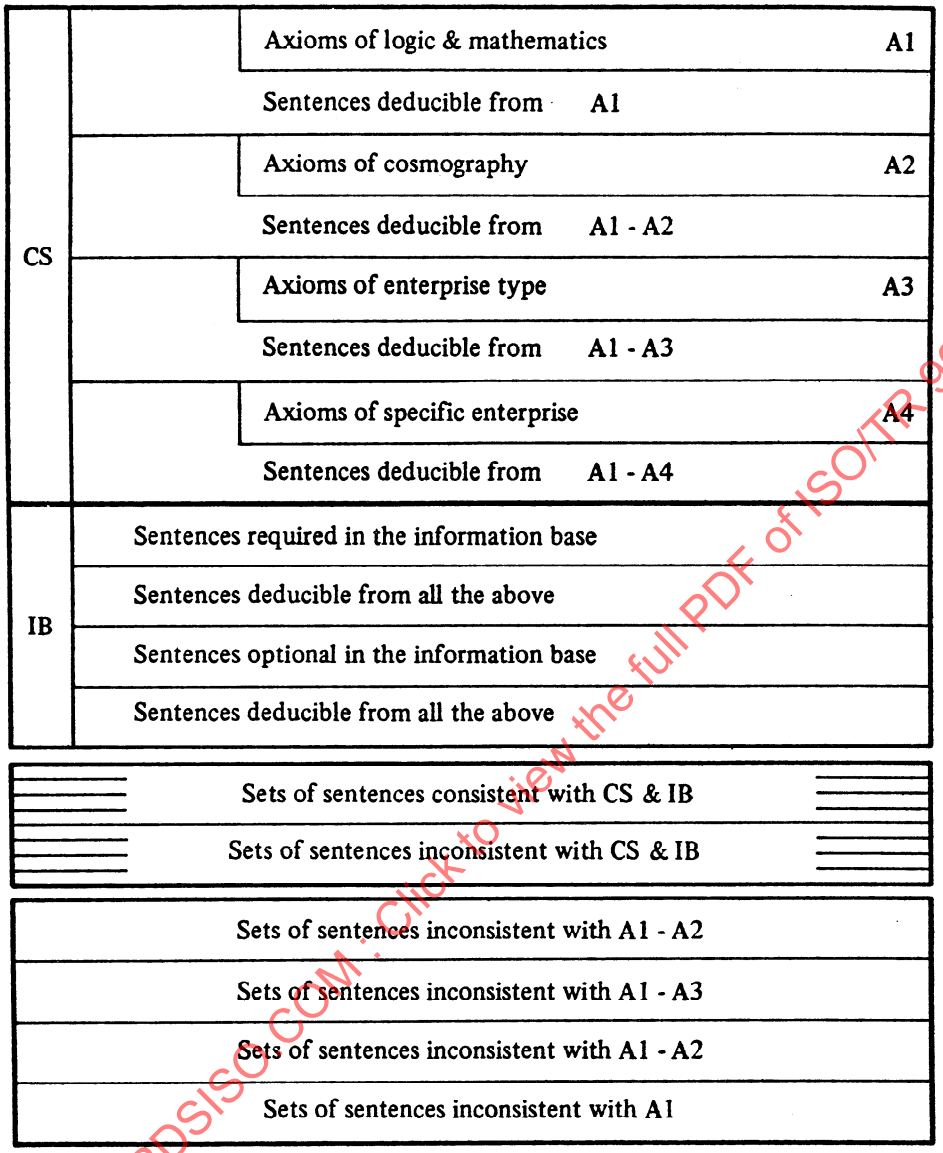


Figure F.3. Classification of L-sentences.

A second set of sentences, A2, somewhat gradiloquently called the axioms of cosmography, will be almost as ubiquitous as A1. These axioms embrace the relevant physical laws pertinant to the universe of discourse under consideration. They will include those axioms invoking the properties of marks and strings, permitting the incorporation of metalanguages into L. In conventional enterprises they will also include the fundamental sentences that deal with such matters as calendrical rules and geographical locations. A1 and A2 taken together will result in the deducibility of an additional set of theorems. It can be expected that both A1 and A2, together with a useful set of their consequent theorems, will be supplied by your friendly vendor.

It is not unreasonable to suppose that there will be a set of constructs

broadly applicable to all enterprises of a given type; e.g., all insurance commpanies, all banks, all universities, etc. Thus there will be a set of sentences, A3, that constitutes the axioms of enterprise type. These axioms may be supplied by vendors as well, or possibly by trade associations. As before, additional theorems are deducible in concert with A1 and A2.

Finally, there will be the set of sentences, A4, the axioms of the specific enterprise, providing the details of the particular enterprise, i.e., the universe of discourse, under consideration. A4 is the set of sentences that will be provided by the enterprise administrator. The totality of theorems deducible from A1/A4 is the conceptual schema for that universe of discourse in an IPL approach.

The conceptual schema does not exhaust the sentences. Among other things, if all possible sentences were in the conceptual schema it would be inconsistent by definition. Further, it would leave nothing for the information base itself to contain. There will, in general, be many different permissible collections of sentences, sets of sentences not inconsistent with themselves or the conceptual schema. Different permissible collections of sentences, of course, may be inconsistent with each other. The information base will consist of one of these permissible collections of sentences together with the theorems now deducible from them in conjunction with the conceptual schema.

It will frequently be the case for a particular conceptual schema and information base that some set of sentences is required to be in the information base. This is a different concept from the axioms or the theorems deducible therefrom. The conceptual schema may require that some parametric value be available, for example. (An instance of this is the need for a sentence asserting the value of fuel consumption in the example universe of discourse of appendix A). Here the conceptual schema does not specify the precise value, merely that there is one. A sentence asserting what that value is must be in the information base but not in the conceptual schema. There may be several possibilities and any of them will do, but one of them must be present.

Given the conceptual schema and the required sentences, the rest of the conceptual schema and information base is optional, providing only that the totality forms a permissible collection. At any given time, of course, the instantaneous state of the conceptual schema and information base is a precisely defined set of sentences.

There will be other sets of sentences consistent with the instantaneous state of the conceptual schema and information base and, therefore, permissible collections of sentences to insert.

There will be other sets of sentences inconsistent with the instantaneous state of the information base, but not with the conceptual schema. Such sets of sentences are permissible collections of sentences with which to modify the conceptual schema and information base by replacing the existing information base.

In addition, there are sets of sentences that are inconsistent with the axioms A4, with A3 and A4, with A2, A3 and A4, and, finally with the entire conceptual schema including A1. Every sentence of L falls into one of the categories considered above, although there are some sentences for which it is impossible in principle to determine into which category they fall.

Given appropriate authorization rules, any set of sentences not inconsistent in itself may replace the set of sentences with which it is inconsistent. If this changes the conceptual schema in any way, a new universe of discourse is being described. This entire discussion is illustrated in figure F.3.

F.5.2. CONSTRUCTS.

The discussion in section F.3.2 was rather sparse. The purpose of this section is to elaborate the primitive concepts into usable forms. First considered are those constructs essential to the basal logic. In this beginning discussion the letters "P" and "Q" will stand in place of arbitrary sentences and the letters "X" and "Y" in place of arbitrary terms. What follows is precise but not fully formal.

Certain connectives are useful and can be defined in terms of the primitive-connectives. These are the usual connectives found in elementary logic:

"(P Or Q)" equivalent to "(If Not P Then Q)".

"(P & Q)" equivalent to "Not (Not P Or Not Q)".

"(P Iff Q)" equivalent to "((If P Then Q) & (If Q Then P))".

One additional quantifier is useful:

"For some X P" equivalent to "Not For all X Not P.

The above four definitions (actually definitional schemata) together with six axiom schemata establishing the properties of:

"Not P"

"(If P Then Q)"

"For all X P"

are sufficient for all logic necessary that is independent of specific predicates.

Five primitive-predicates are required for the present formulation of mathematics, three are singulary and two are binary. Primitive-predicates were introduced syntactically in section F.3.2 as having the form:

"Pr" {lower-case-letter} {prime}.

The "Pr" establishes that the symbol is a predicate, the string of lower case letters merely serve to distinguish one predicate from another and the number of primes determines how many variables must follow the predicate to construct a well formed atomic sentence. Using definitions, more memorable notation can be introduced:

(Null X Iff Pra'X).

(Individual X Iff Prb'X).

(Class X Iff Prc'X).

Interpretation of these three predicates is simple. "Null X" asserts that the entity denoted by X is the null entity, "Individual X" asserts that the entity denoted by X is an individual and "Class X" asserts that the entity denoted by X is a class. In every universe of discourse in the IPL approaches the totality of entities is factored into these mutually exclusive categories. The null entity is the rough equivalent of nothing. The null entity exists, that is there will be a theorem, "For some x Null x", but it is the sort of entity to which things that are impossible reduce, e.g. "Null The square circle"".

Individuals and classes are the interesting entities and everything non-null is one or the other. Individuals are the kinds of entities that are things in themselves and to which the concept of class membership does not apply. Classes, on the other hand, are the kinds of entities to which the concept of membership is central. A class is its members.

Two binary predicates are crucial:

((X = Y) Iff Pra´´XY).

((X Is among Y) Iff Prb´´XY).

"X = Y" asserts that the entity denoted by X is identical to the entity denoted by Y and "X Is among Y" asserts that the entity denoted by X is a member of the class denoted by Y. Obviously Y must denote a class for "X Is among Y" to assert a true proposition but "X Is among Y" is meaningful no matter what Y denotes. This kind of situation is characteristic of the IPL approaches and forces the use of explicit constraints. There must be an axiom such as:

"For all X For all Y (If (X Is among Y) Then Class Y).

The unusual form of the predicate of class member-ship, "Is among", derives from the absence of the Greek letter epsilon in the IS 646 alphabet.

The sufficiency of these primitive-predicates for logic and mathematics has been long since demonstrated (for details see Frege [7], Whitehead and Russell [11], Quine [12], and, for the present context, Steel [17]). Five axiom schemata and seventeen explicit axioms are sufficient in the usual formulations.

The number of definitions that need to be incorporated will depend on the amount of mathematics required for the description of the particular universe of discourse under consideration. In what follows only those concepts relevant to the example of appendix B will be explicated at all and these rather informally, principally to introduce the notation. First, the concept of definite descriptions is:

"The X P" denotes the single entity X such that the proposition asserted by P is true if such a single entity exists and otherwise denotes the null entity.

Descriptions are terms and they are the specific form of term that is defined in the context of all possible positions in atomic-sentences. All other complex terms are ultimately reducible to definite descriptions. Thus, the precise definition of the notion of "the class of all x such that P" or, notationally "{X \ P}" as follows:

({X \ P} = The Y (Class Y & For all X (If For some Z (X Is among Z) Then (X Is among Y) Iff P))).

This definition together with appropriate axioms (not included here) insures that L is extensional, that is that two classes are identical if and only if they have the same members, and that the well-known class paradoxes are excluded.

Classes can be given by exhibition:

$$(\{a,b,c,\ldots,z\} = \{X \setminus X = a \text{ Or } X = b \text{ Or } X = c \text{ Or } \ldots \text{ Or } X = z\}).$$

Ordered pairs can be defined by:

$$([x,y] = \{\{x\},\{x,y\}\}).$$

Relations are then classes of ordered pairs:

$$(\{xy \setminus P\} = \{z \setminus \text{For some } x \text{ For some } y\ (z = [x,y] \ \&\ P)\}).$$

That is, the relation of x to y such that P is the class of ordered pairs [x , y] such that P is true. To assert that A bears the relationship "member of" to B is to assert:

$$([A,B] \text{ Is among } \{xy \setminus x \text{ Is among } y\}).$$

Functions, used in the strict mathematical sense in an IPL approach, are given by:

$$(Fx\ Y = \{yx \setminus y = Y\}).$$

Thus, the function of x whose value (for the argument x) is Y is the relation of y to x such that y = Y. Functions are simply special relations where there is a unique value for each argument. The need to reverse x and y in the definition will not be discussed here. The mutual refusal of nineteenth century mathematicians and logicians to agree on a standard would generate bad language.

For a variety of reasons it is convenient to use a slightly special notation for the result of applying a function. Instead of F(x), L uses:

$$(F{:}x = \text{The } y\ ([y,x] \text{ Is among } F)).$$

If Double = Fx 2x then Double:3 = 6.

A somewhat similar construct permits the identification of the class of all entities that bear the relation R to some member of a specified class. Thus:

$$(R;x = \{y \setminus \text{For some } z\ (z \text{ Is among } x \ \&\ [y,z] \text{ Is among } R)\}).$$

Certain elementary notions of set theory are useful. These are the usual notions of:

Union:

$$(x \cup y = \{z \setminus z \text{ Is among } x \text{ Or } z \text{ Is among } y\}).$$

Intersection:

$$(x \cap y = \{z \setminus z \text{ Is among } x \ \&\ z \text{ Is among } y\}).$$