Information Systems Vol. 8, No. 2, pp. 117–124, 1983
Printed in Great Britain.

0306-4379/83 $3.00 + .00
Pergamon Press Ltd.

SEMANTIC HIERARCHIES AND ABSTRACTIONS IN CONCEPTUAL SCHEMATA

DIRK VERMEIR
Department of Computer Science, University of Queensland, 4067 St. Lucia, Australia

(*Received* 15 *July* 1982; *in revised form* 23 *November* 1982)

Abstract—A conceptual schema is a clear, easy to understand and exact representation of the semantics of an underlying universe of discourse. The role of such schemata in the specification and design of information bases is now widely recognized. However, in "real life" applications, it is often the case that the size of the conceptual schema exceeds the limit above which it is necessary to have an abstraction mechanism available, as otherwise it would become difficult to understand.

In this paper we present several concepts and heuristic procedures which allow for semantic modularization of a conceptual schema, thus providing a concise representation, using different degrees of abstraction and/or viewpoints, of a schema of any size.

1. INTRODUCTION

In recent years, the problem of specifying the information contents of a database system has been widely recognized to be a crucial one in the design cycle[1, 2].

This gave rise to a new discipline: conceptual information analysis or conceptualization which is concerned with the problem of precisely and unambiguously representing the deep semantic structure of a given situation ("Universe of Discourse") in some formal system.

Such a description is commonly called a conceptual schema. One of the most important functions of a conceptual schema is to serve as a specification reference and a tool to communicate the semantics of the information base between all parties concerned.

Therefore, such a conceptual schema should be easy to "read". However, in "real life" applications, it is often the case that the size of the conceptual schema exceeds the limit above which it becomes difficult to understand. Moreover, a particular user is often only interested in a relatively small subset of the conceptual schema or he may not need the maximum level of detail.

Hence the need for semantic "decomposition" and "modularization" of conceptual schemata.

It is useful to note here the analogy between a program and a conceptual schema: programs are split into pieces of manageable size using either top down decomposition ("structured programming") or modularization ("abstract data types") or a combination of both.

For conceptual schemata the situation is more complicated in that a single decomposition is usually not sufficient. Indeed, it should be possible to decompose and/or modularize a conceptual schema from many *different viewpoints*, according to the area of interest of any particular user.

Another important difference, with respect to decomposition and modularization between programs and conceptual schemata is that programming is essentially a constructive process. Indeed, although some aspects of a program are specified beforehand, the programmer is in most cases free to design the program's architecture in a well structured way, chosen by him in advance. On the other hand, modeling a universe of discourse is rather a descriptive activity. The modular and hierarchical structure of the resulting conceptual schema is not chosen by the information analyst. Rather, it is the task of the information analyst to discover the hierarchical and modular structure of a conceptual schema. While it is possible to restructure a program, it is usually the case that one can make only very minor modifications to a correct conceptual schema in order to impose a certain structure.

In this paper, we develop heuristic procedures which allow for the meaningful semantic decomposition as well as modularization of a conceptual schema, from an arbitrary 'viewpoint". The procedures are based on the "binary relationship' formalism for conceptual schemata, as described in e.g. [4]. However the procedures could also be applied on other conceptualization formalisms, e.g. the Entity Attribute Relationship approach and the "nested n-ary relationship" approach. Fortunately, it is possible to automate the procedures so that they may be incorporated, e.g. in an advanced information analysis tool.

The paper is organized as follows: in Section 2 we briefly describe the underlying formalism, i.e. the deep structure binary relationship approach. In Section 3 we describe a method to generate "viewpoint relative" abstraction hierarchies from a conceptual schema while in Section 4 a procedure to generate an "absolute" hierarchical description of the conceptual schema is presented. Section 5 describes some variations and refinements of the algorithms. Finally, Section 6 presents conclusions and subjects for further research.

2. THE DEEP STRUCTURE BINARY RELATIONSHIP MODEL

The deep structure binary relationship model was developed by Nijssen (see e.g. [1]) and others and could be described as a semantic network oriented

approach to conceptual schema modeling. Its main advantages lie in the simplicity of the basic concepts, the strongly typed structure and the careful distinction between "abstract" and "naming" concepts.

A full description of the philosophy and the properties of this approach to conceptual schema description is beyond the scope of the present paper, therefore we only give a short overview of the basic concepts. For a more detailed discussion, the reader is referred to the literature (e.g. [2–4]).

The basic building blocks of the I/B model are:

- The concept of **non-lexical object type** which allows the entities in the Universe of Discourse to be classified.
- Any fact concerning entities can be represented using occurrences of binary relationships or associations (called **ideas**) between object types.

Ideas may be one-to-one functional (many-to-one) or relational (many-to-many). Just as entities are classified into object types facts concerning those entities should be classified into "semantic equivalence classes" i.e. ideas.

An idea imposes two **roles** on the associated object types. In an idea occurrence, each object is involved in exactly one role of the idea. It has turned out to be useful to give separate names to both the idea and its roles.

- The concept of **subtype** allows one to construct complex hierarchies of classification. In addition to its own (discriminating) properties, a subtype inherits all the characteristics of its supertypes.
- Occurrences of object types are rather abstract things: they exist only through their properties (i.e. idea occurrences) which link them to other occurrences. As one eventually wants to reference such things, we need the concept of "self referencing" object type. Such object types are called **lexical object types**. Lexical object types cannot be sub- or supertypes. There cannot be any relationship between lexical object types.
- Binary relations between non-lexical and lexical object types are called **bridges**. Intuitively lexical object occurrences are just "names" which have no semantics by themselves, they are only meaningful in bridge occurrences. Hence bridges should be regarded as naming conventions. A lexical object type can have at most one bridge connected to it.

NOTATION

In the sequel of the paper, we shall denote a conceptual schema as a tuple $S = (L, N, I, B, C)$ where L is the set of all lexical object types. N is the set of all non-lexical object types, including subtypes and subtype links. The sets I and B contain all ideas and bridges respectively. The set C contains all constraint specifications.

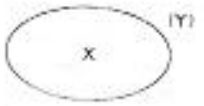
As a shorthand we shall use the term "link" to denote an idea, bridge or subtype link. Also, we will call path any "consecutive" sequence of links between object types. Finally, the term "nolot" will often be used as an abbreviation for non-lexical object type.

Example

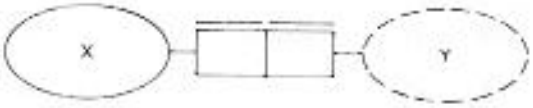
The example conceptual of Fig. 1 illustrates the concepts described above.

Due to space restrictions, not all role names are listed. Note that in the graphical representation, non-lexical object types are represented as ellipses, lexical object types as dotted ellipses. Ideas and bridges are drawn as rectangles (divided into 2 "role boxes") connected to both its "domain" object types.

Functional ideas are indicated by drawing a line over the box corresponding to the domain of the function (one-to-one ideas have two such lines). A dot on a role indicates that the role is mandatory for the corresponding object type. The arc between the associations NR and TYPE indicates that a combination of a SERIAL# and a PLANE_TYPE instance determines at most one instance of AIRCRAFT. Some constraints are not graphically represented, e.g. the constraint which specifies that a STAFF member can only get an ASSIGNMENT on an AIRCRAFT of a type for which he is qualified. Finally, the symbol



is a shorthand for



3. VIEWPOINT—RELATIVE ABSTRACTION AND DECOMPOSITION

In this section we will define the concepts of scope and abstraction. We shall also describe an algorithm which, given a conceptual schema and a viewpoint, will generate a hierarchy of scopes with increasing levels of abstraction.

In the specification of a conceptual schema, all things are "equal" meaning that no object type or idea type is more important than another. While this is ok for small conceptual schemata, say schemata which do not contain more than 20 non-lexical object types, it becomes a problem when faced with large "real-life" conceptual schemata. Indeed, such large schemata are often hard to understand and the semantic links between object types which are "far apart" may not be readily apparent.

The obvious solution to this problem is to use the "divide and conquer" technique to split up the conceptual schema into chunks which have a manageable size. Such a chunk will be called a "scope".

Definition 1

Given a conceptual schema $S = (L, N, I, B, C)$, **a scope** of S is a tuple $S' = (L', N', I', B', C')$ which is such that, for every 2 object types $O1$, $O2$ in S', every link X between $O1$ and $O2$ is also in S'.

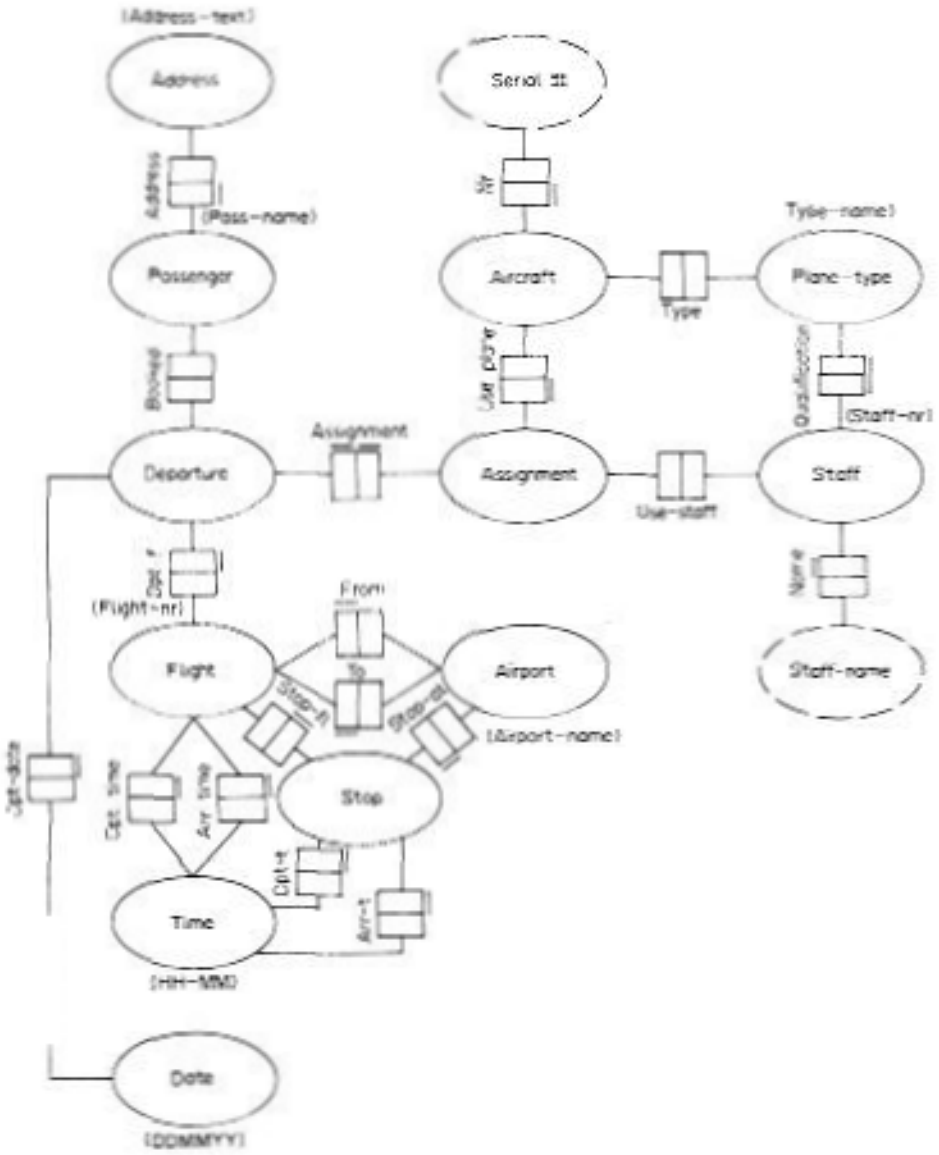


Fig. 1. The example conceptual of Fig. 1 illustrates the concepts described above.

Note that, according to the definition, a scope is just a "complete" subset of the conceptual schema. In this context, we shall use a restricted type of scope, which is called a "connected" scope.

Definition 2

Given a conceptual schema $S = (L, N, I, B, C)$ and a scope $S' = (L', N', I', B', C')$ we say that S' is **connected** if there is a path between every two distinct object types $O1$, $O2$ in S'.

Judiciously splitting the conceptual schema into different connected scopes may then help the user to understand the whole conceptual schema by concentrating on one part of it at a time. The actual definition of the scopes may not be a trivial matter. Indeed, often it is not easy to identify the parts of the conceptual schema which are semantically only "loosely" connected, and therefore eligible to be in separate scopes. Hence the exercise may increase the understanding of the semantics of the whole conceptual schema and its overall structure as well.

Still there are some disadvantages to this approach: first of all, it may be a tedious task to manually split the conceptual schema into the right set of scopes, even with an advanced information analysis system which can store and retrieve scopes. Secondly, in order not to "loose sight" on the interconnections between the various parts, it may be necessary to allow a certain amount of overlap between the component scopes of a conceptual schema. Most importantly, the result is still "flat", i.e. we cannot distinguish between "important" and "auxiliary" concepts (object types).

A solution to those problems lies in the concepts of "key concept", "abstraction" and "abstraction hierarchy". Intuitively speaking, a key concept is an object type which is semantically important because without

it, the conceptual schema would become incoherent (unconnected). Formally we have the following definition

Definition 3

Given a conceptual schema $S = (L, N, I, B, C)$ and a nolot O we call O a **key concept** in S if and only if there exist two other distinct nolots $O1$ and $O2$ in S such that every path linking $O1$ and $O2$ involves O

Example

In the example conceptual schema of Fig. 1, the key concepts are PASSENGER, DEPARTURE, ASSIGNMENT, FLIGHT

The definition above may also be interpreted intuitively as follows: "from the viewpoint of $O1$, the object type $O2$ only exists through O", in other words 'to $O1$ O is the gate to $O2$" or "to $O1$, $O2$ is only an aspect of O' This implies that, if we are interested in $O1$ then we are only marginally interested in $O2$. Thus, from $O1$ we can **abstract** from $O2$, as long as we keep O In such an abstraction on O (from $O1$'s viewpoint), we can drop ("abstraction from") all object types $O2$ which are related to $O1$ only though O

Here is a formal definition of abstraction.

Definition 4

Given a conceptual schema $S = (L, N, I, B, C)$ and distinct object types O and O' of which O is a key concept we call **abstraction on O from the viewpoint** O the connected scope $S' = (L', N', I', B', C')$ which is obtained by deleting all object types and ideas/bridges which are related to O only through O

We say that S is an O-abstraction of all object types in $S - S'$.

It is useful to note here that the concept of abstrac-

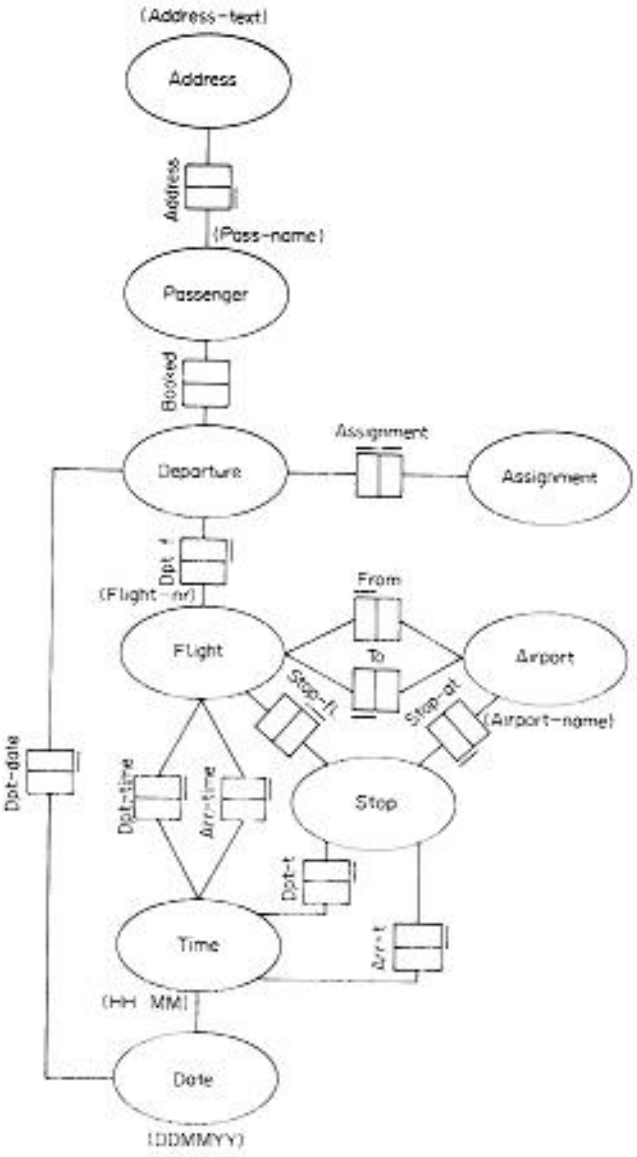


Fig. 2.

tion is symmetric in the sense that if one can abstract of O from viewpoint O then one also abstract of O from viewpoint O' which corresponds to the intuition that 'importance' is often a relative concept, i.e. heavily depends on the point of view one takes (from where one abstracts).

Example

Figure 2 illustrates the abstraction from the passenger's viewpoint on assignment.

From the above it follows that any key concept in a conceptual schema gives rise to two semantically meaningful connected scopes, both containing the key concept. Moreover such decompositions can easily be generated automatically, as the concept of 'key concept' is based only on the formal properties which are known to the information analysis system.

It is our belief that a list of key concepts in a conceptual schema can help to obtain a deeper understanding of the semantics of the conceptual schema. It can also serve as a "debugging' tool. Indeed if unexpected key concepts appear in a conceptual schema then this may indicate that one or more idea links are missing.

As an example consider the following conceptual schema about prospective car buyers. It appears that CAR is a key concept and thus the price of the car may be abstracted of from the PERSON's point of view. However the above conceptual schema would only be 'correct" for rich persons. Most prospective car buyers will have a budget as illustrated in Fig. 4, which effectively prohibits abstraction of the car's price from the PERSON's point of view.

Next we show how one can, from an object type, generate a hierarchy of abstractions which goes from the fully detailed conceptual schema to the "highest possible" level of abstraction.

This is achieved as follows: from a given "focus" object type which the user can select based on his point of view in a conceptual schema, we can partially order all the key concepts in the schema according to their distance" from the focus object type as follows: the closest key concepts (with distance 0) are the ones it is impossible to abstract of. Recursively, if the distance of a key concept to the given object type is greater than i but it can be abstracted of using a key concept at distance i then the distance of this object type is $i + 1$.

We can then formally define a sequence of abstractions $S0$ SN where N is the maximal distance of a key concept from the focus object type, indeed, for a given $i, 0 \leqq i \leqq N$ Si is obtained by abstracting, with the focus object type as viewpoint, on all key concepts which are at distance i from the focus object type.

Thus the algorithm for obtaining the ith level relative abstraction Si from the viewpoint of a nolot O is as follows:

Algorithm 1 *Viewpoint relative abstraction hierarchy*

Input A conceptual schema $S = (L, N, I, B, C)$

(1) Determine the set K of all key concepts in the conceptual schema. (This can easily be achieved using standard algorithms.)

(2) Compute the O-relative distance function dO on K which is defined by

$dO(k) = 0$ if and only if there is a path from k to O which does not contain another key concept (but k and possibly O).

$= \text{MIN}[dO(k')] + 1$ where k is such that there is a path from k to k which does not contain another key concept (but k and k').

(3) Obtain Si abstracting from the viewpoint O on all key concepts k which are such that $dO(k) = 1$

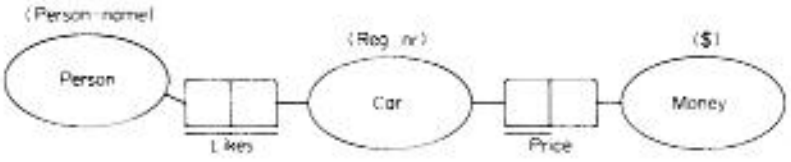


Fig. 3.

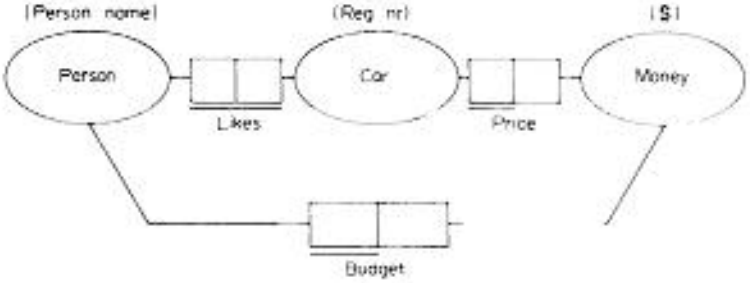


Fig. 4.

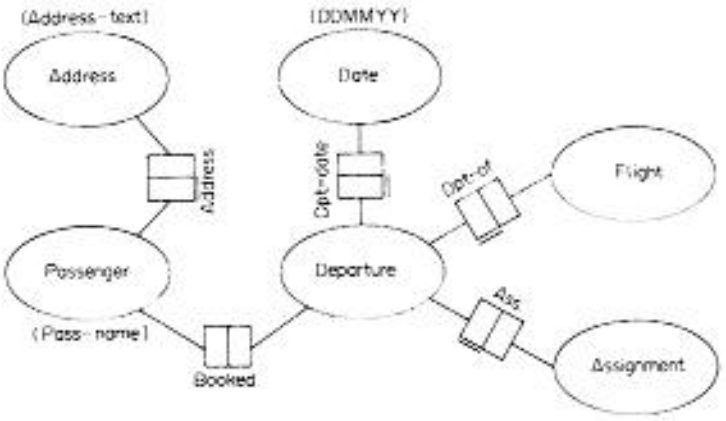


Fig. 5.

Example

Taking PASSENGER as a viewpoint, the distances from PASSENGER to the other key concepts are: DEPARTURE(0), ASSIGNMENT(1), FLIGHT(1).

Figure 5 shows the abstraction $S1$ (level 1) from the PASSENGER viewpoint.

4. AN ABSOLUTE ABSTRACTION HIERARCHY FOR A CONCEPTUAL SCHEMA

In this section, we shall show that the notion of key concept can also be used to derive an "objective", that is unbiased by a particular viewpoint, hierarchy of abstractions from a conceptual schema.

Intuitively, it is clear that the key concepts are the more important object types in a conceptual schema. Thus, the first-level abstraction should contain those key concepts as object types. Additional object types may have to be retained in order to link the key concepts together.

The next higher level in the hierarchy can then be obtained by applying the same procedure to the previous level. This will gradually reduce the size of the abstraction schema as the set of key concepts will strictly decrease with every iteration.

Formally, the algorithm can be described as follows:

Algorithm 2. *Absolute abstraction hierarchy*

Input. A conceptual schema $S = (L, N, I, B, C)$.

To obtain the $i+1$th level abstraction $Si+1 = (Li+1, Ni+1, Ii+1, Bi+1, Ci+1)$ from the ith level abstraction $Si = (Li, Ni, Ii, Bi, Ci)$ (where $S = S0$) do the following:

(1) Determine the set K of all key concepts in Si.

(2) For every nolot O in Ni, put O in $Ni+1$ if and only if either O is in K or at least 2 elements $k1$, $k2$ are at distance 0 from O in Si.

(3) Let $Ii+1$ consist of all ideas which are defined in Si between elements of $Ni+1$. ($Bi+1$ is empty).

Example

Figure 6 shows $S1$ for the example conceptual schema.

5. VARIATIONS

Several variations and refinements on the definitions given in the previous section are possible.

Relaxing the definition of key concept

For example, one could define the notion of key concept on the set of all object types instead of on the set of nolots only. This would make almost every nolot a key concept, as most of them have a bridge to some unique lexical object type, which would be connected to the rest of the conceptual schema only through this nolot. This does not significantly alter the result of the absolute abstraction hierarchy algorithm: it would probably only generate an extra first level abstraction which roughly equals the conceptual schema without the lexical object types and the bridges. The effect on the relative abstraction hierarchy algorithm would be more noticeable: the number of level would probably increase considerably. An undesirable side effect of relaxing the definition of key

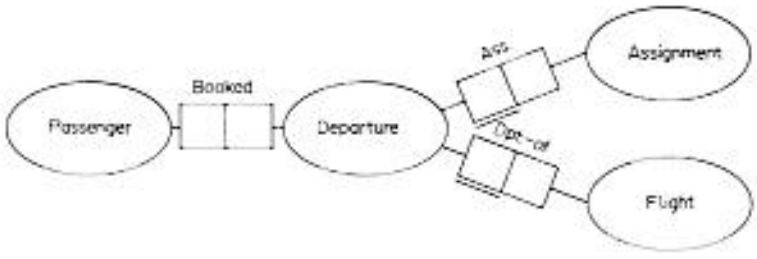


Fig. 6.

concept would be that the notion would become almost equivalent to the notion of nolot, and therefore meaningless. This could easily be remedied by structuring the set of key concepts in a hierarchical manner: the **degree** of a key concept would be equal to the highest level in the absolute abstraction hierarchy where it appears. Using this approach, the key concepts of degree 1 would be roughly the same as the key concepts of the original definition. The solution has the additional attraction of providing a hierarchy of "importance" between object types in the conceptual schema.

Weak semantic connections

A more serious problem with the present approach is that of "weak" semantic connections which sometimes prevent the existence of "natural" key concepts. As an example of this, consider a modified version of the conceptual schema of Fig. 1 where an extra nolot CITY is added which is the range of 2 functional ideas from ADDRESS and AIRPORT respectively. This addition reduces the number of key concepts in the conceptual schema to 1 (ASSIGNMENT), because most nolots are now connected also using a path through CITY. Still, one can observe that the paths through CITY are semantically "weak" because they involve a connection of the type illustrated in Fig. 7.

Connections of this type could be called "common attribute" connections because they relate two object types through a common "attribute". The underlying idea could be paraphrased by claiming a strong semantic relationship between people and rock samples because they both have a weight. Hence an obvious and natural solution to the problem would be not to count paths which contain common attribute connections in the definition of key concept. It should be pointed out here that we do not advocate the deletion of such connections from the conceptual schema. Rather we only want to exclude those connections from playing a role in the definition of key concepts which is only concerned with "strong" semantic relationships.

Another type of "weak semantic connections" arises from the presence of "exclusion" constraints. Intuitively, an exclusion constraint between compatible roles (i.e. roles defined on the same object type or on different object types where one is the subtype of the other) forbids object type occurrences involved in one-role to be involved (at the same time) in the other role. Exclusion constraints can also be defined between subtypes, e.g. as in Fig. 8.

It should then be clear that a path containing 2 roles or subtype links which are involved in a single exclusion constraint should not be counted as a "strong" semantic connection. In the example of Fig. 8, this will allow MARRIAGE to become a key concept.

Limitations

It should be mentioned here that, as we already pointed out in the introduction, the above heuristic

Fig. 7

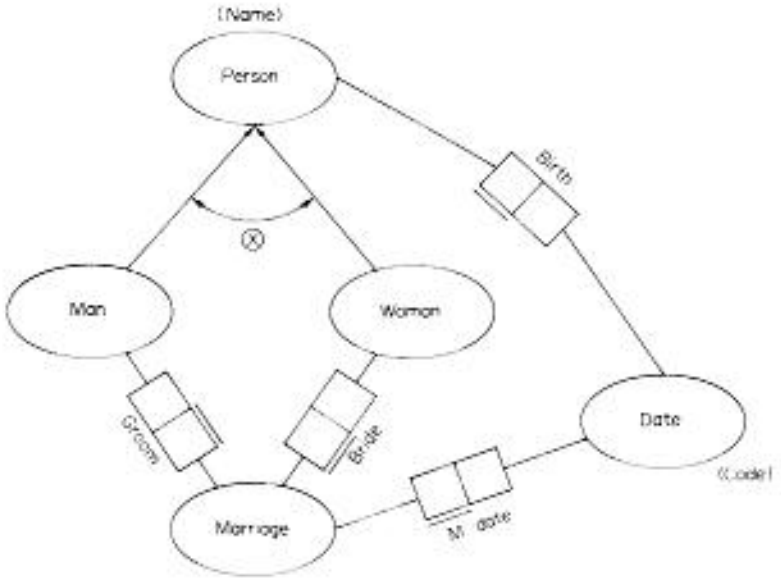


Fig. 8

procedures are mainly intended for use with "large"—say over 40 object types—conceptual schemata. Such conceptual schemata occur frequently in practical applications and, because of their size, they benefit most of the approach described in this document.

Small conceptual schemata are sometimes "strongly connected" and therefore it may be the case that little information is gained from applying the above procedures, due to the fact that there may be no key concepts.

Attribute–hierarchy based abstractions

Another approach to the problem of modularizing and hierarchically structuring a conceptual schema is based on the interpretation of fuctional ideas as generators of an "importance" order between non-lexical object types. The underlying hypothesis is that if a nolot $O1$ is connected to another nolot $O2$ by a (sequence) of functional ideas then $O1$ is assumed to be at least as important as $O2$. This approach is described in [5]. The relation between the present approach and [5] is a subject for further research.

6. CONCLUSIONS

In this paper we investigated the problem of semantic modularization and hierarchical decomposition of conceptual schemata. We presented 2 heuristic procedures which support the automatic generation of both viewpoint–relative and absolute abstraction hierarchies on a conceptual schema. As a by-product, those procedures also yield a partial ordering of the object types of a conceptual schema according to "importance". We believe that these procedures can be useful in an advanced information analysis system to allow for readable and user oriented representations of conceptual schemata.

Several topics remain to be investigated: for example, what is the use of hierarchical decomposition on the generation of a database schema from a conceptual schema. A related topic is the extension of the definition of "key concept" to the population level, which in our opinion may help to decide on optimal physical implementation of the underlying information base. Finally, the present approach could be extended to a more general model which allows for n-ary ideas and "nested" object types.

REFERENCES

[1] G. M. Nijssen: An architecture for knowledge base systems. *Proc. SPOT-2 Conf.*, Stockholm (Sept 1981).
[2] E. Falkenberg: On the conceptual approach to data bases: S. M. Dean and P. Hammersley Eds. *Proc. Int. Conf. on Data Bases*, pp. 92–98. University of Aberdeen, Heyden & Son (1980).
[3] D. Vermeir and G. M. Nijssen: A procedure to define the object type structure of a conceptual schema. *Inform. Systems* 7(4) (1982).
[4] Concepts and terminology for the conceptual schema and the information base. In J. J. Van Griethuyzen (Ed), ISO/TC97/SC5/WG3 Rep. (1982).
[5] A. Amikam: Automatic semantical hierarchies and abstractions. Manuscript, 1982.