Computer Support by Knowledge Enhancement, Constraints and Methodology

de Greef, H.P.

Citation for published version (APA):
Amsterdam: UvA

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 3

Well-Defined Models of Data Sets

This chapter assumes that all objects in the statistical domain are models of data sets. Using the concept of well-defined models, the interrelationships among the objects in statistical tasks are made explicit and a solution is found for consistent modifications of sets of models (i.e., sets of related objects). Having this domain model is an important result in itself, and it serves as an excellent preparation for the StatCons-1 design in the next chapter.

References:


Introduction

Statistical analysis works with data. Most of the work is not in data analysis, but in planning, collecting, cleaning, combining and manipulating data sets, and producing the report. In practice this is no trivial affair. Various directions have been pursued to provide computer support beyond data storage and statistical analysis: statistical expert systems, computerized questionaires, generation of experiment designs to name a few. Support of isolated activities is of little practical help; integration is important.
All activities around data consume or produce information about the data. Aiming at computer support, the question is which information about the data is needed and in which manner this data about the data, hence the term ‘metadata’, can be formalized (De Feber & De Greef; 1992; Van den Berg et al., 1992; Darius et al., 1993; Adèr, 1995; Froeschl, 1997). An important type of metadata is methodological knowledge (Adèr, 1995).

This chapter presents an approach for the representation of methodological knowledge, based on modelling principles similar to those in statistical analysis: information in a data set can often be briefly summarized using a model. There are many readily available languages for models of data sets. Figure 3.1 shows several examples. These models of the data set do not present anything new that is not already present in the data set, but they provide a concise summary of structural characteristics of the data. The data set itself can be regarded as a formal object and there is a formal relation between the language in which a model is formulated, and the data set as a formal object. Models of the data set can be computed from the data set, provided the data are available.

Models can provide a selection of information relevant for a particular activity (user views) by a human or a program. The information needs of the interviewer are different than those of the statistical analyst, and those are different from the needs of the interview program running in the interviewer’s laptop.

Models can also provide abstraction from operational details. In planning and design hierarchies of models can be used to facilitate successive refinement. Models at a more abstract level can be used to formulate requirements and global design decisions. Operational details can be filled-in using models at a more detailed level.

Models of data can provide compressed storage of information present in the data. The compression may be lossless, meaning that no information is lost by the compression. In this way, models can even replace (part of) the data and be used to store data efficiently. For example, if all cases in a data set are male, then the data set may contain a variable ‘sex’ that has the value ‘male’ for each unit. Instead of this the variable ‘sex’ can be left out of the data set (the entire column can be deleted), and be replaced by a model for this part of the data, for example the expression ‘sex=male’. This saves storage space, it is a form of lossless data compression. Special attention is needed if the data set is modified. If data of women is added the model is not valid anymore, and it changes to ‘sex in [male, female]’. This may be an indication that the variable ‘sex’ has to be stored explicitly into the data set itself.

Models of methodological aspects can be defined in a similar manner. The usual concept of data set can be extended, to include more information about methodological aspects in the data set itself. For example, with the notion of time, as in Langefors’s (1977) elementary datum: (object, property, value, time). These additions to the elementary datum need not be stored in any real data set; models may be used to efficiently represent and store the additional information about the data in a compressed manner.

The term ‘data set’ is usually associated with the concrete result of data collection: the file with data to be subjected to statistical analysis. A data set in this chapter is more of a theoretical object. Most, if not all, additions to the elementary datum are not intended for use in real data sets, but to provide a basis for models.

Secondly, a data set may also serve as a representation for plans. If a researcher and
a statistician have a meeting to discuss plans for an investigation there is no data set yet, but they are already talking about a data set. In a platonic sense this data set already exists, that is what researcher and statistician apparently assume. It is only that the thing, or a sample thereof, yet has to be collected. If this conversation establishes a list of variables and a sample size, the contours of the data set have already become established.

During the discussion, in an iterative design process, the plan or the platonic data set can be changed repeatedly. In information systems that support such work processes, such modifications are difficult. If the problem representation is changed in one place, for example, the value set of a variable is modified, then the problem representation may need to be changed in other places as well. The issue is that a modification involves much more than just editing. Compare this to changing a text. If all occurrences of 'he' are changed to 'they', then many other changes (e.g., the verbs), must be made to keep the text consistent. There is a need for a knowledge representation that supports such changes and helps to maintain consistency. This is a general requirement for all computer support: metadata, including methodological knowledge, should be adaptable (Adèr, 1995).

This chapter presents well-defined models as basis for representations that can be
modified in a consistent way. There is a central, platonic data set; all objects in statistical tasks are models of this central data set. Modifications in the problem representation are modifications of these models, and, if we take care that all models are well-defined, then an entire collection of models (i.e., the problem representation in its entirety), can always be modified in a consistent way.

The use of well-defined models of data provides a framework for metadata with many desirable features. It provides an unambiguous meaning to metadata. It facilitates the expression of structure in the data. It makes interrelationships among various metadata elements explicit, and it allows for consistent modifications of metadata.

Section 3.1 introduces models. Section 3.2 defines a language for classical data sets: \(\{(\text{unit}, \text{variable}, \text{value})\}\). Section 3.3 presents the idea of well-definedness in an informal manner. Section 3.3.1 defines well-definedness in a mathematical framework, that is, as a homomorphism from data set language to model language. Properties of homomorphisms provide a general solution for consistent modifications. Section 3.4 investigates interrelations among well-defined models of a common data set. It provides glimpses of the structure that exists among homomorphic images of a common data set.

Then an example is given of language design using the well-defined models concept in the design of representations for data collection plans. Section 3.5 extends the data set concept with time (3.5.1), and operational definition (3.5.2). Then examples are given of well-defined models for the temporal aspects of the data collection process (3.5.3). The final section (3.6) can only conclude that well-defined models provide a firm and indeed well-defined basis for metadata languages. They can be consistently modified, are defined over the entire domain of data sets, and seem to have many more desirable properties.

### 3.1 Models

All important theories of models have in common that there are two worlds between which there is a correspondence that is not so much in the elements—the worlds are different, the elements are not identical—but that is in the structure, allowing one to say something about the other. *Modeling Theory* (Rosen, 1991), provides a concise and elegant definition.

A model requires two systems, interrelationships among these, and three criteria to be met. The first criterion is that each system has elements and relations and an entailment function. Entailment means that there are laws governing the system. Events or situations in the system have a necessary consequence or correlate. To give an example, in a formal logical system the proof procedure and proof rules provide entailment (Socrates being human, and humans being mortal, entails Socrates being mortal, using the modus ponens rule). In a physical system causation, for example, may be taken as entailment.

The second criterion is that there are encoding and decoding functions between these systems. Elements and relations in one system can be mapped to the other system, and back. The context so far is illustrated in Figure 3.2. This figure also shows the third criterion. The diagram of functions is a commutative diagram. Starting from a \(w1\) in \(W1\), there are two paths to get at \(w1'\). Both paths lead to the same result.
entailment encoding decoding entailment

W2 is a model for W1 if for all w1 in W1
f(w1) = decoding(g(encoding(w1)))

Figure 3.2: Modeling theory.

If this third criterion is fulfilled, the second system can be said to be a model of the first. I find this formulation from modeling theory a simple and precise formulation of the essence of model theory as can be found in, for example, Genesereth & Nilsson (1987), and mathematical modelling as described by Maki & Thompson (1973). Below such view on what a model is will be used, but impregnating it with concepts from statistics (Table 3.1), and making it more precise using the concept of homomorphism rather than isomorphism.

<table>
<thead>
<tr>
<th>System1:</th>
<th>The data set as a formal object (e.g. equation 3.1).</th>
</tr>
</thead>
<tbody>
<tr>
<td>System2:</td>
<td>A model for the data set stated in a particular language</td>
</tr>
<tr>
<td>Entailment1 (f):</td>
<td>An operation on the data set</td>
</tr>
<tr>
<td>Entailment2 (g):</td>
<td>An operation on the model statement</td>
</tr>
<tr>
<td>Encoding:</td>
<td>Computational methods of a wider class than, but including those as provided by statistical packages</td>
</tr>
<tr>
<td>Decoding:</td>
<td>Generation of a data set that satisfies the model</td>
</tr>
</tbody>
</table>

Table 3.1: Introduction to the well-defined model concept.

### 3.2 The Data Set

The data set is the central object in a statistical investigation. A data set is a record of data collection in a sample of units. It may be a record produced by interviewers going door to door with a questionnaire obtaining answers from respondents. Or it may be the result of different groups of patients being treated with different combinations of drugs and their progress being monitored. Both statistics and data base theory provide formalizations for storing the information.

\[(\text{unit, variable, value}).\] Let us start from the data set that is most often used in statistical investigations: a rectangular array (matrix, table, spreadsheet) with rows for Units \((\text{i.e., Cases})\), and columns for Variables. A complete rectangular data set contains a value for all pairs \((\text{unit, variable})\) in \(\text{Units} \times \text{Variables}\). Table 3.2 provides
Table 3.2: Example of a complete rectangular data set of 5 units by 3 variables.

<table>
<thead>
<tr>
<th>unit</th>
<th>sex</th>
<th>maritalStatus</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>person1</td>
<td>female</td>
<td>mar</td>
<td>29</td>
</tr>
<tr>
<td>person2</td>
<td>female</td>
<td>unmar</td>
<td>36</td>
</tr>
<tr>
<td>person3</td>
<td>female</td>
<td>mar</td>
<td>28</td>
</tr>
<tr>
<td>person4</td>
<td>male</td>
<td>mar</td>
<td>35</td>
</tr>
<tr>
<td>person5</td>
<td>female</td>
<td>unmar</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 3.2: Example of a complete rectangular data set of 5 units by 3 variables.

an example of such a data set. But, in a data set, not all units need to have values for all variables. Reasons for that may differ, for example, the data has not yet been completely collected, or not all variables are defined for all units, or data collection happens in a non-systematic way. In the framework above, we cannot describe this situation, unless we introduce special values. For example ‘...?’ for a value not yet known, and ‘not applicable’ if the variable is not defined for the unit. Table 3.3 provides an example.

Table 3.3: Example of an incomplete rectangular data set.

<table>
<thead>
<tr>
<th>unit</th>
<th>sex</th>
<th>beard</th>
<th>nOfPregnancies</th>
</tr>
</thead>
<tbody>
<tr>
<td>person1</td>
<td>female</td>
<td>-not applicable-</td>
<td>2</td>
</tr>
<tr>
<td>person2</td>
<td>male</td>
<td>yes</td>
<td>-not applicable-</td>
</tr>
<tr>
<td>person3</td>
<td>female</td>
<td>-not applicable-</td>
<td>...?</td>
</tr>
</tbody>
</table>

Table 3.3: Example of an incomplete rectangular data set.

To decrease the tension between a rectangular format and a possibly arbitrarily shaped data set, we rather use the following formalization: an elementary datum is a tuple \((unit, variable, value)\). A Data Set is a set of such elementary data. This provides a simple formalization of the data set:

\[ DataSet = \{(unit, variable, value)\} \tag{3.1} \]

Because variables are functions (Hays, 1974), there is a constraint upon the Data Set: there is not more than one tuple for each \((unit, variable)\).

\[ DataSet \subseteq (Units \times Variables \rightarrow Values) \tag{3.2} \]

The subset relation indicates that not every data set must be a full cross product of \(Units\) and \(Variables\). The cross product establishes the maximal domain for the \(DataSet\). The domain of an actual data set is either this cross product or a subset. Table 3.4 shows a data set structured according to this format. Data that are missing or not applicable are simply left out. This formalization is attractive, because the elementary datum can be extended with additional information about the valuation.

**Operations on Data Sets** The Data Set is a dynamic object, not only because new data can be inserted, but also because it can be modified. The formalization chosen makes it simple to define operations that change the Data Set. Set theory provides two basic operations on a data set: inserting and deleting an elementary datum. Using these, it is not difficult to define larger operations. For example: delete variable or delete units that satisfy a certain criterion.
<table>
<thead>
<tr>
<th>unit</th>
<th>variable</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>person1</td>
<td>sex</td>
<td>female</td>
</tr>
<tr>
<td>person1</td>
<td>nOfPregnancies</td>
<td>2</td>
</tr>
<tr>
<td>person2</td>
<td>sex</td>
<td>male</td>
</tr>
<tr>
<td>person2</td>
<td>beard</td>
<td>yes</td>
</tr>
<tr>
<td>person3</td>
<td>sex</td>
<td>female</td>
</tr>
</tbody>
</table>

Table 3.4: Example of a data set in (unit, variable, value) format.

### 3.3 Well-Defined Models of a Central Data Set

Models of the data set are descriptions, representations, summaries, or specifications of some aspect of some part of the *DataSet*, as defined in equations (3.1) and (3.2), or according to the more extended definitions to follow.

There are numerous languages for such models available (*e.g.*, Figure 3.1). These languages have a formal semantics, that is, there are clear formal criteria to establish correspondence between the data set and a model expressed in the language. For these models correspondence can simply be based on a function from data sets to models.

![data set compute model](image)

This is not to say that such a model always will be the result of a computation from a data set. Like the drawing of a house, the model may precede the thing being modelled (design of a new house) rather than the other way around (drawing of an existing house). The function is just a convenient way of expressing the relation between data set and models. This function may have an inverse: the data set can be computed or generated from the model. If it does not have an inverse, there can still be a function or procedure that generates data sets that are in the relationship.

**Corresponding operations.** In the world of data sets, it is possible to define a small set of basic operators that can be used, in composition, to transform any data set into any data set.

Using the terminology of software packages for statistical analysis, the most important operations are:

- Deleting a variable (column).
- Inserting a new variable (column). This can be a completely new variable or a new variable as a function (i.e., computation) of existing variables. Important examples of the latter are recoding of a variable or summation of a number of variables to obtain a total score.
- Deleting or selecting units (rows) using a certain criterion that can be applied to each unit (row).

These are similar to Codd’s (1979), projection operator (delete variable), the extension operator (insert variable) and restriction (select units) operator. Such an operator transforms a data set into a new data set.
If such an operation is performed on the data set, then it may be that an operation has to be performed on various models of this data set. Similarly, if an operation is performed on a model, then perhaps something should change as well in the data set or in other models.

**Well-defined Models of Data Sets.** A model of the data set is well-defined if and only if the following commutative diagram holds:

\[
\begin{array}{c}
\text{data set} \\
\downarrow \text{operation} \\
\text{data set'}
\end{array}
\]

\[
\begin{array}{c}
\text{compute} \\
\text{model}
\end{array}
\]

\[
\begin{array}{c}
\text{data set'} \\
\downarrow \text{compute} \\
\text{model'}
\end{array}
\]

By defining for each operation on data sets the corresponding operation on a model, the entire system of the data set and the entire system of the model are brought to a correspondence relation.

**Consistency Between Well-defined Models.** If there are different models for the data set, we can define corresponding operations for each different model. For example, if we have two different well-defined models, we can for each separately predict what they would be like after a certain operation on the data set.

\[
\begin{array}{c}
\text{model}_1 \\
\downarrow \text{compute} \\
\text{data set} \\
\downarrow \text{operation} \\
\text{model}_1'
\end{array}
\]

\[
\begin{array}{c}
\text{corresponding} \\
\text{operation}_1
\end{array}
\]

\[
\begin{array}{c}
\text{data set} \\
\downarrow \text{compute} \\
\text{data set'} \\
\downarrow \text{compute} \\
\text{model}_2
\end{array}
\]

\[
\begin{array}{c}
\text{corresponding} \\
\text{operation}_2
\end{array}
\]

\[
\begin{array}{c}
\text{model}_2' \\
\downarrow \text{compute}
\end{array}
\]

If the data set is not available, consistency between well-defined models can be maintained, provided the models are consistent to begin with. Once all models have been defined in relation to the data set, the actual data set is not needed.
By adhering to strict criteria for a well-defined model of the data set, the interrelations between different models in different languages are (made) tractable. The data set itself need not to be available and the corresponding operations can be trusted to maintain correspondence between models under various operations.

**Schemata for Data Sets** An example of a model of the data set is its schema. Figure 3.3 shows specimens of four different schema specification techniques all applied to the same data set in the middle of Figure 3.3. The examples illustrate how various schema languages may be regarded as functions from a common domain of data sets. Secondly, with this example it is easy to see how various operations on data sets have their counterparts in each schema language.

In the writing above, in the examples, and in the presentation of the 'commutative diagrams', no distinction was made between a pair (dataset, sentence-in-a-language), and the general case: the domain or universe of data sets $D$, and all sentences in the domain of sentences $M$. The general case is: all pairs $(d, m)$ in $D \times M$.

The story of this section can be retold with greater precision, using mathematical language, and with greater impact, by addressing the general case, after having recognized that mathematical concepts such as isomorphism and homomorphism can be used.
3.3.1 Structure-Preserving Functions

The idea of a mapping or function \( f : A \rightarrow B \) from one set to another is very general. If \( A \) and \( B \) have some internal structure, defined by operations upon them so that they can be indicated by \([A, \Diamond]\) and \([B, \Box]\), then the interesting functions are the ones that preserve that structure. Functions that do so are homomorphisms, with isomorphisms as a special case.

**Homomorphism** When \( A \rightarrow B \) is a homomorphism, the effect of the operation in \( A \) is preserved in \( B \). That is, applying first the operation in \( A \) and then \( f \) gives the same result as applying \( f \) first and then the operation in \( B \). This is expressed as \( f(\Diamond a) = \Box f(a) \), or by stating that the following diagram is commutative:

\[
\begin{array}{ccc}
A & \xrightarrow{\Diamond} & A \\
\downarrow f & & \downarrow f \\
B & \xleftarrow{\Box} & B
\end{array}
\]

Similar definitions can be formulated for binary operations, for external operations (presented below), for operations in many-sorted algebras (cf. Dougherty and Giardina, 1989; Vickers, 1989), and so on.

**Surjective Homomorphism** A surjective homomorphism is an onto function (i.e., a surjection). The onto property ensures that for all \( b \in B \) one can choose an element \( a \in A \) such that \( f(a) = b \). This means that if there is a problem of the type: "What would be the effect of applying \( \Box \) to an element \( b \in B \)?", the answer can be obtained by going from \( B \) to \( A \) by choosing an \( a \in A \) with \( f(a) = b \). To this \( a \) the operation of \( A \) can be applied and the result can be mapped by \( f \), producing the required answer in \( B \). With a surjective homomorphism, \( A \) can be said to simulate \( B \).

In addition to the commutative diagram of the homomorphism, the following diagram also holds:

\[
\begin{array}{ccc}
A & \xrightarrow{\Diamond} & A \\
\downarrow \text{choose an } a & & \downarrow f \\
B & \xleftarrow{\Box} & B
\end{array}
\]

**Isomorphism** When a (surjective) homomorphism is isomorphic, \( A \) and \( B \) can simulate each other. The success of the two-way simulation depends on an additional property of the isomorphism: \( f \) is a bijection (an onto function that is one-to-one). Then, \( f \) has an inverse \( f^{-1} : B \rightarrow A \) that is also an isomorphism. When \( A \) and \( B \) are isomorphic, the diagram of the homomorphism holds, and the following two commutative diagrams hold as well:
If there is a surjective homomorphism but not an isomorphism from $A$ onto $B$ (the function is many-to-one), then $A$ simulates $B$, but $B$ only imperfectly simulates $A$; that is, $B$ is isomorphic to a structure composed of classes of $A$. Each class is the inverse image of an element of $B$.

**Homomorphism with respect to two external operations** Among definitions of several different homomorphisms provided by Bouvier & George (1983), there is one that matches the definition of well-definedness of models of data sets. It is the definition of a homomorphism with respect to two external operations: Let $[D, \Diamond]$ and $[M, \Box]$ be sets with an external operation of which the common domain of operators is $\Omega$. The operation $\Diamond$ applies an element of $\Omega$ to an element of $D$ and returns an element in $D$. The operation $\Box$ applies an element of $\Omega$ to an element of $M$ and returns an element in $M$. A function $f : D \rightarrow M$ is a homomorphism from $[D, \Diamond]$ to $[M, \Box]$ if for all pairs $(\alpha, d) \in \Omega \times D$, $f(\alpha \Diamond d) = \alpha \Box f(d)$. That is, the following diagram commutes:

$$\begin{array}{ccc}
\Omega \times D & \xrightarrow{\Diamond} & D \\
\downarrow{id_{\Omega} \times f} & & \downarrow{f} \\
\Omega \times M & \xrightarrow{\Box} & M
\end{array}$$

This is a more precise version of the diagram used in the definition of a well-defined model. Now let $D$ be a set of data sets (i.e., the powerset of $DataSet$), and $M$ a model language, for instance, tables with counts. The set of common operators, $\Omega$, contains a few basic operators, for example 'delete variable', with the name of a variable as parameter. The operation $\Diamond$ applies it to the data set. In a data set with rows for units and columns for variables, an entire column is deleted. The operation $\Box$ applies 'delete variable' to the table of counts. The table is collapsed over that variable, and numbers of units are added to produce the counts in the collapsed table.

More models can be introduced, for example, let $[M_2, \Box_2]$ be a language for schemas for the data set. The operation $\Box_2$ applies the common operators to a schema. Limiting oneself to models that are homomorphisms of a central data set brings the advantage that entire sets of different models $[M_i, \Box_i]$ can be modified in a consistent way, using the common set $\Omega$ of basic operators. Implementation of more complex operators using $\Omega$ operators just once then provides an overall implementation in all models available and in those yet to come. Page 59 provides an example.

Summarizing, well-defined models are homomorphisms of the data set and this guarantees preservation of the effect of data set operations in operations on models in model languages. Therefore, the operators preserve consistency under modifications.
Table 3.5: Summary of the well-defined model concept.

This well-defined models concept provides a solution to the problem of consistent modifications of representations. Everything that is contained in a well-defined model is also contained in the data set. As long as we limit ourselves to well-defined models, an entire set of statements about a data set can be consistently modified using the common operators. A summary of this concept is provided in Table 3.5.

### 3.4 Interrelationships Among Models

The previous section defined the meaning of language expressions in relation to a data set. Using this we can study the interrelationships between different languages, that is, different well-defined models. First the concepts of homomorphism and isomorphism are used to analyze this structure. Second, models, as statements in model languages, are compared on the basis of their extension in the universe of data sets.

**Homomorphic and Isomorphic Functions Between Models.** Even with a simple data set concept, there are many languages that can serve as well-defined model. Among these, many functions may exist, and some of these functions may be homomorphic and even isomorphic. These properties can be used for a classification of functions in three categories: isomorphic, homomorphic but not isomorphic, and non-homomorphic.

Among the schema languages indicated in Figure 3.3 there seem to exist a few homomorphisms, but on closer inspection, and trying to generalize from statements to languages, they appear to be ‘almost homomorphic’ more often than not. The classification using homomorphism and isomorphism as criteria has a disadvantage: pairs of models that are very similar and pairs that appear totally unrelated, end up on the same pile of non-homomorphic pairs of models. Below we indicate means to analyse this pile in more detail.

When the purpose is not to analyze a set of existing languages, but when the aim
is to design one's own set of languages, it is possible to select a set of model languages that are homomorphisms of one another. When designing one's own, one can start from a data set concept that contains the information of interest, and then define a non-cyclic series of homomorphic functions: \( \text{DataSet} \rightarrow M_1 \rightarrow M_2 \rightarrow M_3 \ldots \), that filter and select the information in steps. The composition of two homomorphisms is also homomorphic. Therefore, when introducing a new model, it suffices to show there is a homomorphism from one of the previously established models to the new model. Such a chain of levels of specificity can be useful in planning and design of a data set. Planning and design may traverse the series in the inverse direction, starting with models of low specificity, which are refined to models of higher specificity. Along the way, according to the previous section, the set of models may be modified repeatedly, using the common operators.

**Equivalence and Specificity** A sentence \( m \) in a model language \( M \) is a statement about a data set. A data set satisfies the statement or does not satisfy the statement. The statement \( m \) thus partitions any 'universe' of data sets, that is, \( D \), into data sets that satisfy the statement, the extension, and data sets that don't. To give a simple example, the language statement: 'number of units = 100' may be regarded as a well-defined model. It can be defined as a function from data sets, the number of units is simple to assess. The set of all data sets with 100 units is the extension of the statement.

The relationships between models can be studied as relationships between extensions. If we compare the extensions of a pair of statements in different languages there are four possible outcomes (Figure 3.4).

Two statements \( m_1, m_2 \) in different languages are equivalent if and only if their extensions are identical. Then they state exactly the same about the data. The languages \( M_1 \) and \( M_2 \) in which they are expressed are different, but if their extensions are identical, the models are interchangeable. If this can be generalized, the languages are equivalent. In that case there exists an isomorphism between the languages.

Of two statements \( m_1 \) and \( m_2 \) in different languages the first is more specific than the other if the extension of the first is a subset of the extension of the second. If this can be generalized to all pairs of statements and data sets, language \( M_1 \) is more specific than language \( M_2 \), and there is a homomorphism \( M_1 \rightarrow M_2 \). The data set itself (or a copy thereof), may be regarded as the most specific model of itself. If a data set satisfies the more specific statement of a pair, it also satisfies the less specific statement. There can be series of models of increasing specificity, that is, a series of homomorphic functions.

In the two other cases (at the bottom in Figure 3.4), the two extensions cannot be compared. If any function exists at all between these model languages, it certainly isn't homomorphic. When the extensions do not overlap (the third case in Figure 3.4), there is no data set that satisfies both. The statements are incompatible or inconsistent. When the extensions overlap (the fourth case in Figure 3.4), it is always possible to construct a compound statement of greater specificity by taking the conjunction of two intersecting models.

Using conjunction, one can engineer levels of increasing specificity by starting with a model that will have the role of least specific model, and introduce other models only in conjunction with the least specific model (\( M_0 \)).

Introducing models in conjunction with another model can also help to smooth
Figure 3.4: Comparing Extensions. Each subfigure shows for two statements (i.e., 'models') their extensions in the domain of data sets.

out differences between languages that are almost equivalent. The most obvious candidate for $M_0$ is the overall schema, that is, the variables and their domains. That is, when $m_1$ and $m_2$ are not equivalent, $m_0 \land m_1$ may be equivalent to $m_0 \land m_2$, and it may be possible to generalize this over the languages. The schema defines a more local 'universe' in which the similar models/languages are in a homomorphic or even isomorphic functional relation. Since in any practical application there will be a schema, no harm is done to general applicability.

The concepts of equivalence and specificity (in analogy with isomorphism and homomorphism), and the possibility of taking the conjunction of two models provide us with the means to find some structure in the chaos of potentially available languages for models of data sets. We can now put models in equivalence classes that are ordered with respect to specificity.

To give an example of statements in different languages that are equivalent when each is taken in conjunction with the schema, the following schema for the data set is assumed:
The following four statements in different languages all have the same meaning in terms of the data set, provided they are taken in conjunction with the above schema. They all express the same property of the data set; they are equivalent. The first is a table with counts with the cell count in one cell set to 0.

<table>
<thead>
<tr>
<th>Age</th>
<th>Marstat</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 16</td>
<td>unmar</td>
<td>...</td>
</tr>
<tr>
<td>&lt; 16</td>
<td>mar</td>
<td>0</td>
</tr>
<tr>
<td>&gt; 16</td>
<td>unmar</td>
<td>...</td>
</tr>
<tr>
<td>&gt; 16</td>
<td>mar</td>
<td>...</td>
</tr>
</tbody>
</table>

The second is a table structure, that is, an enumeration of the cells not set to 0.

<table>
<thead>
<tr>
<th>Age</th>
<th>Marstat</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 16</td>
<td>unmar</td>
</tr>
<tr>
<td>&gt; 16</td>
<td>unmar</td>
</tr>
<tr>
<td>&gt; 16</td>
<td>mar</td>
</tr>
</tbody>
</table>

The third is a table formula defining a cross product with one element deleted.

\[ \text{Age} \times \text{Marstat} - (\text{age} < 16, \text{marstat} = \text{mar}) \]

The fourth is a logical expression:

\[ \text{age} < 16 \rightarrow \text{marstat} = \text{unmar} \]

The most striking difference is the difference in size or sheer volume occupied by the language expression.

**Level of Compression** If two statements are equivalent they express the same information. If one of the two is smaller, for example measured by the number of bytes required for storage, it can be said to provide a compressed representation. Among equivalent statements, compression is lossless. At each level of specificity the equivalent statements, and perhaps the languages, can be (partially) ordered according to size or compression level. This gives a 'two-dimensional' classification. In Figure 3.5 it is used for the examples above.

Generally speaking the size or compression dimension may be problematic. For instance, with compression methods used in computer graphics (Gomes & Velho, 1997), there are conditions or situations in which the more compressed language is not defined. This cannot happen with well-defined models, since well-defined models are defined over the entire domain of data sets. Secondly, there are conditions in which there is no reduction in size or where the compressed version is larger than the original. The reason is that the length of a language expression also depends on what is being expressed. Generalizing the "more compressed" relation from statements to languages can be difficult. It will seldomly be the case that one language is consistently better for all data sets. Compressed languages are often designed with a particular situation in mind, (i.e., certain symmetries or certain invariants in the data set), and
provide good compression in this situation. Otherwise they may fail to provide size reduction.

Bearing these limitations in mind, the two-dimensional lay-out is useful in clarifying the interrelations among languages.

**Presentation** There is still variation among languages that are equally specific (i.e., equivalent), and that are of the same level of compression. Even these may look remarkably different. For example, a table with counts may be presented in a cross-tabular format:

<table>
<thead>
<tr>
<th>Age</th>
<th>Marstat</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 16</td>
<td>unmar</td>
<td>...</td>
</tr>
<tr>
<td>&lt; 16</td>
<td>mar</td>
<td>0</td>
</tr>
<tr>
<td>≥ 16</td>
<td>unmar</td>
<td>...</td>
</tr>
<tr>
<td>≥ 16</td>
<td>mar</td>
<td>...</td>
</tr>
</tbody>
</table>

These statements are equivalent and, approximately, of the same size. The remaining difference can be characterized as one in presentation.

The example of the cross-table fails in the generalization from statements to languages. The cross-table simply is not defined on the entire domain of data sets, and therefore it cannot act as a language for well-defined models. Consider the operation of adding a variable, it is not possible (excluding possibilities of an N-dimensional lay-out). There are however, many other languages, including graphical languages, that can act as well-defined models. But even at this point in the classification, were we have defined equivalent models of the same level of compression, the pile of languages can still be quite large.

**Summary.** This section introduced a few concepts and means that can help in the analysis and design of languages for well-defined models. One clear result is a simple recipe to create a cascade or hierarchy of well-defined models:
1. Define a function by which the statement of the new model can be computed, either directly from the data set or from another well-defined model. The function must be defined over the entire domain.

2. Define the application of the $\Omega$ operations in the new model.

3. Show that the function is a homomorphism.

The result from the previous section still stands. Even if interrelationships between model languages can be hard to characterize, models can always can be consistently updated using the $\Omega$ operations.

The next sections will explore this well-defined models concept in the design of representations for statistical and methodological knowledge.

### 3.5 Extending the Elementary Datum

By enlarging the concept of data set, more information about the data collection process, for example Ader’s (1995) methodological knowledge, can be stored in the data set itself. This extended data set concept can then serve as a common basis for various well-defined models that represent statistical and methodological knowledge. In normal everyday data-analysis practice the extra properties of the elementary datum need not be included in real data sets. They serve a theoretical purpose, to provide a basis for well-defined models.

Potentially there are many properties one could add to the elementary datum. Here we concentrate on a representation of the data collection process. That is, the data set is to be extended to capture a trace of the data collection actions. Each datum represents action in the real world involving a unit and the researcher. When we extend the elementary datum with *time*, the data set will contain the timing and ordering of all the actions. When we also add operational definition, the elementary datum will contain a precise action specification.

With these additions, the data set itself contains a trace of the actions of the researcher, and can serve as a base language (*i.e.*, a saturated model), for the data collection process. If data is collected in a systematic way, according to a plan or design, there will be many invariants and symmetries in the extended data set. In this situation, the structure of data collection can be expressed in a more succinct fashion using a few well-defined models. If data is collected in a haphazard, chaotic fashion, most models will fail to provide any compression. Then one can always use the data set as a base language.

#### 3.5.1 Time

Values for variables of units may change in time and there are data sets were time is an integral part, as shown in the example of Table 3.6. In this example, each variable is observed at two time points.

This is an example of a data set in which time is very prominent. In many investigations and data sets time is not prominent, but time or time order is always of great importance in research design. First of all, the ordering of observations of an individual unit is considered important for causal reasoning. For example, the meaning of a variable (*e.g.*, the state of the patient), measured before or after an experimental
Table 3.6: Example of a data set that explicitly takes the time of observation into account.

treatment (e.g., the drugs given to the patient) differs considerably. In a survey with a questionnaire ordering is also important. The interviewer must know in which order to ask the questions. In general one has to know the order in which different variables are observed with each individual unit.

The elementary datum can be extended with a time property. We adopt Langefors’ (1977) definition of an elementary datum as a tuple: \((object, property, value, time)\). To adapt to everyday parlance in the statistical domain, a property is called variable and an object is called unit. This motivates the following definition of a Data Set:

\[
DataSet = \{(unit, variable, value, time)\}
\]  
(3.3)

To express the constraint that each unit can at most have one value for each variable at each time, the definition can be narrowed down to:

\[
DataSet \subseteq (Units \times Variables \times Times \rightarrow Values)
\]  
(3.4)

The subset relation indicates that not every data set must be a full cross product of \(Units, Variables\) and \(Times\). The cross product establishes the maximal domain for the \(DataSet\). The domain of an actual data set is either this cross product or a subset.

Table 3.7: Example of a data set with timestamps.

There are choices in what we take as domain for time. Table 3.7 shows three options. One option is time as measured on clock and calendar. Then it becomes a timestamp for the elementary datum the moment it is observed and recorded in the Data Set, assuming observing and recording happens at the same time. However, when duration is irrelevant and only the sequence of time points is important, we may simply use rank numbers. If we only need to express the sequence for each unit
separately, we may start for each unit with time is 1 (cf. Ader, 1995). The exact time when a unit is engaged can be expressed at the level of the unit, that is, as another variable in the data set.

Since the Data Set will serve as the central, most specific model, the time scale should be at the most specific level. Data sets in which this time scale is mapped onto a less specific time scale may be regarded as well-defined model of the central data set.

In the elementary datum, time is the time when the elementary datum is observed and inserted into the DataSet:

\[ \text{DataSet}_{\text{time}} = \text{DataSet}_{\text{time-1}} \cup \{(\text{unit, variable, value, time})\} \]  

(3.5)

First of all there is a concern that such valuation is true to reality. The valuation provided by the elementary datum must correspond to the valuation in reality at time. Having a tuple such as: \((\text{patient23, bloodpressureH, 112, 17})\) in a data set means that it has been observed that patient number 23, for the function bloodpressureH, maps to a value of 112, at time = 17. To ensure this the researcher, or one of his helpers, must operate or interact with the unit in the real world at time. Therefore, the meaning of the elementary datum has two aspects: a certain valuation at time and a certain action at time.

With a so-called retrospective measurement (e.g., a question to the respondent in a survey about events in the respondent’s past), the time of the observation and the time of the valuation may be different. For example, in an interview, it is asked whether the respondent was subjected to a tonsillectomy, and, if applicable, some questions around this event are asked. The sequence of events \((i.e., \text{birth, tonsillectomy})\), and the sequence of questions do not need to be the same. Both sequences may be important, and, if so, then two time properties are needed. Here in this chapter the emphasis is on the process of data collection. Therefore time is the time of the action by the researcher or one of his helpers.

If existing data sets or archives are being used in an investigation, the time of the original collection and the time of the action by the current researcher are different. Since in this chapter the emphasis is on the process of data collection viewed as actions or interactions with the unit, we only use a single time property in the elementary datum, for the timepoint at which the original observation was made.

### 3.5.2 Operationalization

The time property puts an important aspect of data collection in the data set itself: the temporal order in which the lab assistant or the laptop must collect the data. To provide a more complete instruction for the data collection, the actions and instruments involved must be specified.

Given an elementary datum, for example: \((\text{patient23, bloodpressureH, 112, 17})\), someone could ask serious questions about the ‘bloodpressureH’ function and one would want transparency regarding the method of establishing the value (questions about the scale, the manufacturer of the instrument, the tolerances, and so on). According to scientific standards, the method by which the value is observed should be described in sufficient detail such that others (or at least investigator’s peers) can replicate it in their laboratories. This is referred to as the operational definition or
operationalization: the meaning of a value is in the details of the procedure by which it is obtained.

The operationalization gives information about the action the researcher or his helper must perform. It ought therefore be included in the elementary datum, to provide a more complete representation of the plan or process of data collection in the data set itself.

\[ \text{DataSet} = \{(\text{unit, variable, value, time, operationalization})\} \quad (3.6) \]

Information about the operationalization also contributes to the meaning of the valuation \((\text{unit, variable, value})\). The domain of operational definitions cannot be defined in advance. It is up to the creativity of the researcher and the customs in the research area. However, it is possible to use a global classification. Just like the time scale may be abstracted to simple time ranks, the domain of operationalization may be abstracted to three types. Following the distinctions made in the theory of block designs, and in the GENSTAT system (Nelder, 1974), there are three types of action that can ensure that an individual \text{unit} has a certain value for a \text{variable} at time. These can be introduced by example:

With measurement, patient23 arrives, the measurement device is applied, and its value is read. It reads 112. The researcher can look at the clock, and insert the datum with the right timestamp in the data set. With treatment, patient23 arrives, is hooked-up to a measurement device, and given some drug until the device reads 112. Then, after consulting the clock, the researcher can insert the datum with the right timestamp in the data set. The unit, the variables and their specific values are all determined beforehand, but the researcher must go through the motions to make them true. With sampling/selection, the investigator walks around, measuring different patients, until one is found that gives a reading of 112. This one is selected and will serve as patient23. The researcher can look at the clock, and insert the datum with the right timestamp in the data set. Variables and specific values for the selection are determined in advance.

<table>
<thead>
<tr>
<th>Time</th>
<th>Actions by the Researcher</th>
<th>Insertions in the DataSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sample a unit (unit63)</td>
<td>(unit63,sex,male,1,sampling/selection)</td>
</tr>
<tr>
<td></td>
<td>with sex=male</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>treat unit63</td>
<td>(unit63,drugA,yes,2,treatment)</td>
</tr>
<tr>
<td></td>
<td>with drugA=yes</td>
<td>(unit63,drugB,yes,2,treatment)</td>
</tr>
<tr>
<td>3</td>
<td>measure unit63</td>
<td>(unit63,bloodpressure1,.r1,3,measurement)</td>
</tr>
<tr>
<td></td>
<td>bloodpressureH=..</td>
<td>(unit63,bloodpressure2,.r2,3,measurement)</td>
</tr>
</tbody>
</table>

Table 3.8: Example of a data set as a trace of the data collection process.

These examples show that the same elementary datum can have very different meanings indeed. It can be a value that has been measured, freely allowed to take any value that it might take, or it can be an action specification for selecting or
treating a unit as to make it have a certain value. The three categories are assumed
to be exhaustive, that is, there is no other action or method to obtain an elementary
datum through interaction in the real world with a unit. However, in a data set one
may find elementary data of a different origin. For example, an elementary datum
may be computed from others. In that case the the new datum may inherit the
classification of its sources, provided it is the same category for each of the original
data. The timestamp of the computed variable is the same as the most recent one
among the original data.

Summary. The elementary datum with \textit{time} and \textit{operationalization} provides us with
a data set that represents the data collection process. The data set becomes a trace
of the actions of the researcher. Table 3.8 gives an example, showing how operations
in the real world may correspond to operations on the data set. More concise or less
specific languages for the data collection process can now be studied as well-defined
models of the data set in equation 3.6 above.

It is possible to analyse the interaction between researcher and unit in more de­
tail. Chapter 6 provides techniques for modelling interaction between agents such as
researcher and unit. In this chapter we do not analyse \textit{operationalization} in such
detail, we only use the three types of operational definition as elementary processes
or action primitives that leave their trace in the data set.

Extending the data with time, and having well-defined models thereof, warrants
additions to the set of \( \Omega \) operators. The operators for adding/deleting a variable
will have a version that takes a pair \textit{variable} and \textit{time} as parameter. The effect of
add variable is that \textit{variable} is inserted at \textit{time}. The effect of delete variable is that
\textit{variable} at \textit{time} is deleted.

\subsection{3.5.3 Well-Defined Models for the Process Structure}

If the data set language can describe a trace, it can also describe a plan for a certain
trace in the future. In this plan one may use timestamps in the future, and use
placeholders for the values to be measured in the future. Well-defined models can
then be used as more concise representations of this plan. Examples of this dual
usage can also be found among the classical statistical models. For example, a table
with counts may be used to describe the statistical design before data collection and
it may be used in statistical analysis, where the counts are computed from the data
set as collected. Below we discuss a few well-defined models for the \textit{time} property.

The process structure describes the timing of the actions in data collection and
this temporal structure can be discussed at different levels of specificity. The data
set with \textit{time} on a date and clock scale serves as the most specific level. Using this
as a basis, we can define homomorphic functions that hide unnecessary details and
concentrate on the information that is relevant. This provides a number of levels of
decreasing specificity. Figure 3.6 shows a diagram. The data set is shown at the
bottom, to the right.

Starting from the data set, each abstraction step is based upon an assumption.
First it is assumed that \textit{value} and \textit{operationalization} are not important in the
temporal structure: therefore these properties of the elementary datum can be ig­
nored. Then to provide a more compressed representation, for each unit the pairs
\((\textit{variable}, \textit{time})\) are collected in a set.
Next it is assumed that only ordering is important: therefore, one may use ranks or timepoints instead of timestamps. In addition it is assumed that units are independent and that only the temporal ordering within a unit is important. Therefore the time property can be mapped to unitrank. This leaves us with \{\text{(unit, [(variable, unitrank)])}\} as a more abstract language.

For each unit, the set \([(variable, unitrank)\)] shows one or more variables at each timepoint unitrank, and the set can be compressed to a sequence. Using the example data in Table 3.8, the sequence of unit63 is the following:

\[
\text{sex} < \text{(drugA, drugB)} < \text{(bloodpressureH, bloodpressureL)}
\]

Determining the sequence for all units gives a set \{(unit, sequence)\}. Sequences can be counted, showing the number of units following the different sequences. When the interest is not on the numbers of units, but on the set of sequences that exist in
the data, the counts can be ignored. This gives the set of sequences \( \{(sequence)\} \). It contains the sequences present in the data.

If the sequence is invariant over units (i.e., all units go through the same sequence), as it will usually be the case in a well-designed investigation, this 'set-of-sequences' language provides a rather good compression: a set with one sequence. When the invariant isn't perfect, when there are different sequences for different units, the set will contain all these sequences. All the variants (i.e., the second sequence, and more if there are more), are reported. There is less compression, but no information is lost.

The abstraction and compression can be carried on a little further. For example, by simplifying to \( \{\text{(variable, unitrank)}\} \), showing the timepoints of each variable. The timepoints of each variable can be collected in a set: \( \{\text{(variable, [unitrank])}\} \). As soon as there appear two rank numbers or time points for one variable, it is a sign that one may need a more specific language. There are different situations that can give rise to a variable occurring at two time positions: multiple instance of the same variable in a single sequence, or different sequences among units. To differentiate, one needs a more specific language, and that would be the 'set-of-sequences' language. There is nothing wrong with the ranks language, it is still well-defined, but for tasks that need detailed temporal information, it lacks specificity when the process structure is complex.

Summary. In the design of languages for well-defined models, one can build a cascade or tree of homomorphisms, with decreasing level of specificity, for a particular aspect of the data set. There is another characteristic of well-defined models of which the importance is becoming more and more recognized: Well-defined models are defined for the entire domain of data sets, meaning that they can always be computed. They are simple and robust, they will always do their job, and always have a clear and unambiguous meaning. When there are imperfections of the invariant or symmetry (assumptions are violated, lack of fit), the models are still defined, but they provide less compression, and relevant information may be lost. Then a more specific model in the cascade or tree may provide the specificity needed. If all fails, there is always the data set itself, as the most specific language.

Complex operations example. Having these abstractions of the temporal structure may suggest certain complex operations on the data set. For example, a swap operator might be considered useful. It swaps timepoints of two variables, and its action can be defined using the common operators. Having done that, the swap operator can be effectively be applied to the data set with time stamps, and to any other well-defined model in which time is present. This example is to show that models may suggest more complex operators, that the set \( \Omega \) of common operators is in no way limiting, these common operators can be used to define the complex operator, and, having done that, the complex operator can be applied to all well-defined models.

### 3.6 Conclusion

The starting point of this chapter is that all objects that are worked with in statistical tasks are models of data sets. A model is formulated in a language, and there are many such languages for models of data sets available, to an amount that is confusing.
The result of this chapter is the concept of well-defined model. It has great advantages to use well-defined models.

**Ω-Operations** If one needs representations that can be modified consistently at any time, well-defined models provide the solution. Because well-defined models are formal, they provide a representation with a clear and unambiguous meaning. The interrelationships among different models and languages can be traced and can be made explicit through the data. Consistent modifications of models can be based on corresponding modifications of the data. If the data set is being described by a set of different models, the set of models can be consistently modified by applying to each model the corresponding operation for that model. The corresponding operations thus provide a solution for consistent modifications. It appears possible to define a set Ω of common operators that can be used to modify the data set and sets of models in a consistent manner.

**Ordering by Specificity and Compression.** Secondly, we can see glimpses of the structure among well-defined models of a common data set. This brings some order in the confusing multitude of languages available. With well-defined models we can order models on specificity, and, for equivalent models, we can order languages on degree of compression. In the next chapter, the two-dimensional ordering diagram will be used often. Operators such as conjunction can help to make interrelations among models that are similar but not equivalent more explicit.

We may want to minimize the number of different languages, and pick just one. For example, from the most specific languages one may choose the one that is most compressed. Nevertheless, we may still want to use more than just one language, because languages may differ in level of difficulty for the user, or for other reasons as mentioned in the introduction of this chapter. The concept of well-defined models does not prescribe the selection criteria, but it greatly facilitates selection because it provides a sorted overview of options.

It is also possible to simultaneously use a number of different models that address the same aspect of the data at different levels of specificity. The common operators will take care of consistent modifications of any set of models.

**Extensible** The concept of well-defined models of data sets is extensible. A larger data set concept allows a richer set of well-defined models. One example was given of an extended data set concept and a model that addresses time and data operations: the process structure. Other properties of the data set and other models can be introduced when a particular application requires these. Time and data operation is what we will need in the next chapter for well-defined models in statistical design.

Well-defined models provide a radically general and robust solution for consistent modifications at a fundamental level, and there are, I expect, much more interesting properties and implications of well-defined models. Being fundamental, well-defined models do not provide all the answers in the design and implementation of systems that use metadata. Below a few adjoining issues are briefly mentioned.

**Meta Classes.** A theme that recurs again and again is that a formalism in itself does not tell enough. The sentences of a well-defined model have a clear
meaning, namely that the data set has a certain property, but this is not enough. However, in the actual use of such sentences there is an additional meaning that is not given in the formalism itself. The same model of the data set (e.g., identical formulae or expressions), may have different meanings, depending on the role in the task. An example for statisticians is statistical analysis. The same ANOVA formula or expression can have the role of Maximal Model, Alternative Model, or Best Model (Wolstenholme & Nelder, 1986).

Another example occurs in statistical design. The same formula or expression or table can have the role of Requirements, Design Solution or Constraints. Think of the example with a zero in a table with counts presented earlier. It could be the law of the land, and then it is not possible to have that data, therefore, in a design task, the table or expression would have the role of Constraints. It could also be the decision of the researcher and then it belongs to the Design Solution. Alternatively, it might be part of the Requirements, as a Maximal Model for the statistical analysis.

The knowledge-engineering approach KADS (Schreiber et al., 1993), stresses the importance of meta-logical classifications based on the role in the task or in the problem solving process. This concept of meta class is also applicable to well-defined models. In models of statistical problem solving, these meta-classifications can be made explicit—each statement is associated with its role, or statements in different roles are stored separately—and can be used to control the problem solving process.

Extending the elementary datum with the operationalization can also be thought of as providing a meta-classification. The elementary datum with time can have different roles in the data collection process, and the operationalization property tells which role.

**Meta-Data Management.** Well-defined models are relevant for data/meta-data management systems. The project ‘Modelling Metadata’ (Darius et al., 1993), was based on pairing a data set with its ‘metadata set’. The project investigated the use of binary operations to merge and align data/meta-data sets. A full functional prototype demonstrated the feasibility of this approach. A recent thesis on meta-data management is provided by Froeschl (1997), and it contains many pointers to ongoing meta-data research.

**User-Interface Design.** Well-defined models are important for user-interface design. The selection of representations to be used in the interaction with the user can be made with more deliberation, to allow ease of learning and ease of use. The availability of operations is a good basis for a more flexible dialogue in which the user does not have to redo previous work, or make errors in editing, whenever a change is made.

**Interoperability.** Well-defined models also provide a basis for interoperability of different software systems. Many investigators today have to use at least two or three software systems in their work, and any modification that is desired may force the investigator to go back to each of the three systems. Sharing a common set of operators $\Omega$ would provide integration and reduce a change to a single operation that updates all well-defined models across different software systems.
In conclusion, well-defined models provide a firm and indeed well-defined basis for meta-data languages. They can be consistently modified, are defined over the entire domain of data sets, have a clear meaning, and seem to have many more desirable properties and open up a whole new universe for further study. Well-defined models will be used in the next chapter in the knowledge acquisition for a support system for iterative statistical design.

The next chapter contains a section with another more extensive example (section 4.2), of the use of well-defined models in the analysis and design of statistical software.

Acknowledgements. Many thanks are due to Joachim Mowitz and David Hand who provided valuable comments on an earlier version of this chapter. This research was partly performed within DOSES project B 41: “Modelling Metadata”, funded by EUROSTAT.