Journal of Official Statistics, Vol. 28, No. 3, 2012, pp. 413–440

The Organization of Information in a Statistical Office

Tjalling Gelsema[1]

A theoretical framework for statistical data and metadata is presented using well-known techniques from computer science. It is shown that many kinds of statistical information are best represented by a function. The relationships between different pieces of statistical information, such as the dependency between aggregated data and the underlying microdata, are explained using transformations of functions. A modest set of such transformations is considered and the identities that hold between them are shown. Thus they describe a class of algebras, and, according to the theory of initial algebra semantics, the initial algebra in the class is the natural candidate for recording metadata.

Key words: Metadata; aggregation; semantics; information modeling.

Introduction

This article aims to combine results and techniques from computer science theory with the efforts and the objectives of the statistical metadata field.

One of the challenges National Statistical Institutes (NSIs) face is to cope with huge amounts of statistical information. While obvious for large amounts of data, the raw material of the statistical production process, this also holds true for statistical metainformation. Because of the wide range of different social phenomena addressed, an NSI is increasingly dependent on the proper management of metainformation to assist in the retrieval, the understanding and the processing of statistical data. To claim that an NSI's main asset is not its data, but the knowledge to interpret that data is therefore less controversial than it seems on first thought.

In a similar way that a pharmacist relies on a well-organized cabinet to store and retrieve its medicines, proper metainformation management depends on a well-structured model. To extend the pharmacist analogy, such a model lays down the organization of the drawers in which different pieces of information will be placed, such as variable descriptions, elements of classification structures, estimation procedures etc.; in short: all the information that is collected during the design phase of a statistical production process.

The reality within many NSIs though is that during the design phase such information becomes scattered across many different information systems which seem hard to integrate, and as a result of which the chance of information divergence is substantial.

[1] Statistics Netherlands, Henri Faasdreef 312, 2492 JP Den Haag. Email: te.gelsema@cbs.nl
Acknowledgments: The author is grateful to Erik van Bracht for his careful review of an earlier version of this article, and for his many useful comments. Also, the author received many instructive and detailed comments on Gelsema (2010) from two anonymous references. These comments were highly appreciated during the completion of this article.

For instance, within Statistics Netherlands (SN), the identification and nomination of intermediate and output variables is handled through an information system, the main task of which is to aid the design of a statistical process. The description of such variables however is stored via SN's Data Service Center (DSC), a separate information system for the office-wide storage and retrieval of SN's statistical data. Worse, one of the products of process design closely related to the identification of variables is the so-called domain model: a model expressed in the Unified Modeling Language (UML) of the statistical domain under consideration, which is usually documented separately (in document form) and which is used in the design of statistical databases that handle the local storage of data.

Each of these three ways of storing information uses a different technique and language to capture the information, taking its own point of view as predominant. With the exception of the second (the information model supporting the DSC), whether it is a language for process design, the UML, or a modeling technique in the Entity Relationship (ER) style, none of them is developed specifically with the capturing of statistical information in mind and each has therefore ambiguous semantics of typical statistical notions, such as the concept of aggregated data. For instance, in the ER case, a relation with entities *activity*, *size class*, and *turnover* containing information about enterprises can be interpreted as an aggregate — in which case *activity* and *size class* form the dimensions of the aggregate and *turnover* is interpreted as a total or an average— or it can be interpreted as a microdata set — in which case *activity*, *size class* and *turnover* describe the result of measurements from each enterprise individually. Because of these ambiguities, exchange of statistical information between information systems is cumbersome and hence system integration becomes difficult to achieve.

Even more, despite efforts such as the infological approach (Sundgren 1973) the problem of "reality → data mapping", which we interpret as the translation between the conceptual design of a statistical domain of interest and database design, is still approached in an ad hoc fashion. While members of the first discipline speak of variables, populations, class parameters and classification systems, members of the second use entities, attributes, data sets and records to express themselves; the exact translation between the two frames of reference is seldom enunciated however. This is especially manifest for aggregation, the process of transforming data about individual entities of interest into data about classes of entities, the core business of an NSI nota bene. In this respect it is remarkable that while NSIs have become increasingly dependent on relational databases for the processing of data, notwithstanding the fact that most commercial database query languages support various aggregation operators, aggregation was excluded (and still is, to our knowledge) from the theory (Codd 1969; 1970) upon which today's relational database management systems rely.

Existing information models that do focus on capturing statistical information are often problematic for other reasons. Of these, we can mention the Statistical Data and Metadata eXchange initiative (SDMX 2011), a standard for the automated exchange of aggregated statistical data, and the Common Metadata Framework (CMF 2011), a model that supports information systems managing statistical variables and classification systems. They fail to meet the following four requirements (and, in fact, we know of no other model capturing statistical information that meets them all). While the argument is made against SDMX, similar arguments can just as well be contrived against CMF.

Clear relationship between data and metadata: In SDMX, the (structural) metadata for an SDMX *Data Set* is given by a so-called *Data Structure Definition* (*DSD*, also called *Key Family*). Though the relationship between a *Data Set* and its *DSD* is given on SDMX (2011, p. 77) the class diagram that captures it is complex, comprising approximately 30 classes and 40 relations between them. This is probably due to the fact that data can be described in four essentially different ways: by a *PrimaryMeasure*, by a nontime *Dimension*, by a time *Dimension* and by a *DataAttribute*, respectively. While XML schemas exist that, given a *DSD*, validate a *Data Set*, checking these schemas for correctness is problematic due to the complex relationship between data and metadata. SDMX does not provide correctness proofs of their XML schemas.

Correct semantics: SDMX does not give a clear understanding of basic statistical concepts, such as the concept of a variable or the mechanism of aggregation (in fact, the term variable is encountered only twice in the documentation (SDMX 2011)). Instead, SDMX seems to be designed according to what is believed to be aggregated statistical information based on examples frequently encountered in official statistics. Thus SDMX could be called an *inductive* model, which can lead to incorrect semantics as we will see shortly, rather than a *deductive* model. As an example, because classifications in NSIs often exhibit a tree-like structure, SDMX describes them as hierarchical. In contrast, the observation that a classification is meant to subdivide a population into classes naturally leads to the use of Boolean algebras for that purpose — because a finite Boolean algebra *is* the structure of a finite set of subsets of a population— which are richer than hierarchical structures. In fact, it is easy to imagine a classification structure that is not hierarchical (Gelsema 2008) and, in particular, the regional classification used by SN is an example of a nonhierarchical classification. This is because water board districts often cross province and region borders.

Avoidance of synonyms: In SDMX basically equivalent data sets can be described in a variety of essentially different ways. Of course, this counteracts the use of SDMX in providing standards for aggregated data exchange. One example in which SDMX stimulates the occurrence of synonyms is the distinction between time series and cross-sectional data: an arbitrary data set can be described using either format. Also, irrespective of format, a *DSD* designer has a choice in which he can describe multiple aggregated variables either as separate *PrimaryMeasures*, or he can use a single *PrimaryMeasure* (usually labeled *OBS_VALUE* in that case) and put in an extra *Dimension* to differentiate between the variables. How to transform *Data Sets* between either modes (in a general sense) remains an unsolved problem.

Avoidance of nonsense: Finally, a *DSD* designer meets few constraints in constructing and describing an SDMX *Data Set*. Thus it becomes possible that, for instance, a *Concept* (say NACE) used to designate a *Dimension* in one *Data Set* can be reused to designate a *PrimaryMeasure* in another. Also, any choice of *Dimensions* and *PrimaryMeasures* can be combined in a *DSD*, even if this makes no sense with respect to statistical content. Thus, e.g., *total turnover of classes of enterprises grouped by sex* does not meet any complaint as far as SDMX is concerned. Of course, nonsense (especially nonsense from the point of view of statistical content) cannot always be avoided through an information model (and our model is no exception). However, especially when large collections of metadata need to be maintained and might be chosen from when designing a statistical data set, some restrictioning based on statistical content is required.

It seems fair to say then that today's techniques of modeling statistical information are not in optimal shape. What we propose in this article as a remedy is a framework for statistical information that meets the above requirements and that (unlike the models above) is guided by techniques well-rooted in computer science. The strategy to construct such a framework reads as follows.

i) Give a natural and precise (mathematical) meaning to basic concepts involving statistical data, such as the concept of a statistical variable;
ii) Give natural meaning to derived concepts, such as that of a microdata set and an aggregated data set. Give proper preconditions to avoid constructing data sets that make no sense;
iii) Identify and prove equalities between (different) constructions of identical data sets;
iv) Finally, let the metadata of a data set be one specifically selected construction (a so-called normal form term) among all (equal) constructions of that data set.

Techniques from computer science enter the scene in the fourth step above, where the precise (mathematical) relationship between statistical data and metadata is given.

Formal semantics (Goguen et al. 1978; Wirsing 1994) has been a discipline within computer science theory for decades and has been successful in particular in the field of programming language design (Gunter 1992). Of particular interest is *initial algebra semantics* (Goguen et al. 1977; Goguen and Malcolm 1996), the relevance of which for statistical metadata we will sketch next. For completeness' sake, we give the definition of an initial algebra from Goguen et al. (1977): in a class of algebras $\mathcal{D}$, an algebra T is initial if for every D in $\mathcal{D}$ there exists a unique homomorphism $h_D : T \rightarrow D$. This will only be used tacitly in the sequel however.

In a statistical office, two of the functions of statistical metadata are to describe and to retrieve statistical data. Putting these to work, we imagine having at our disposal an office-wide mapping h that takes a metadata term t — expressed in some format or (formal) language — and that returns the data item (the data set, or the variable, or the statistic) d of interest that is described by the term t. Thus, we envisage that through h we find the data $h(t) = d$ that corresponds to the metadata term t, as for instance suggested by the expression $h(\textit{GDP in bln. euros}) = 415$. At SN such a mapping is currently in operation: it is the office-wide DSC mentioned earlier.

For simple deterministic data manipulations, such as aggregation or the 'column-wise' combining of two data sets, one could argue that the metadata of the result of the manipulation should completely (and automatically) be determined by the metadata of its argument(s). Thus, e.g., if $h(\textit{GDP in bln. euros}) = 415$ and $h(\textit{debt as a \% of GDP}) = 68$, and if $\langle 415{,}68 \rangle$ is the column-wise combination of the two statistics, then we can conceive a corresponding manipulation $\langle \textit{GDP in bln. euros, debt as a \% of GDP} \rangle$ on their metadata terms such that $h(\langle \textit{GDP in bln. euros, debt as a \% of GDP} \rangle) = \langle 415{,}68 \rangle$. This suggests that h is a homomorphism from metadata terms t to data items d, since then we have $h(\langle t_1,\ t_2 \rangle) = \langle h(t_1),\ h(t_2) \rangle$ in general. Further, it makes sense to demand that if two metadata terms t and t' are equivalent — in the sense that they use two different ways of describing the same thing — then $h(t)$ and $h(t')$ should point to the same data. For instance, one could require that $h(\textit{debt as a \% of GDP})$ and $h(\langle \textit{debt as a \% of GDP} \rangle)$ (a single column) should both be 68. In a nutshell, these requirements portray the initial algebra approach to formal semantics translated to the domain of statistical metadata.

Initial algebras have many desirable properties, we can just mention here that 1) they are basically unique, 2) they have a canonical representation (the so-called term algebra, or rather, the algebra which we obtain from the term algebra if we identify terms that can be transformed into each other, given a set of equations) and 3) the homomorphism h sketched above is immediate once we have a view of our manipulations ("zap!", according to Goguen et al.). The first property is desirable because it takes away any unnecessary discussion about which metadata model suits best: basically, there is just one. The other two imply that we may concentrate on the *intended semantics* (Goguen et al. 1977), i.e., step (i) through (iii) of our strategy — the initial algebra, if it exists, along with our map h, then follows more or less immediately, which completes step (iv). The existence of the initial algebra is shown in Appendix A, where the material developed in this article is translated to the framework of equational logic (Meinke and Tucker 1992).

So we now turn to our view of statistical data. The simple idea developed here is that statistical information, throughout the process from data collection to publishing, is of a functional nature. For instance, a series of numbers that is the result of a measurement (taking the variable *number of years employed in current job* as an example) on a population of 6 employees and informally denoted by

$$V = 6, 4, 7, 4, 3, 7$$

is more accurately represented as a function $v : p \rightarrow x$, with p the population of employees and x a set of possible outcomes. If we let the set $p = \{e_1, \ldots, e_6\}$ be (a representation of) the population, and we take $x = \{0, \ldots, 60\}$, then the variable V above is given by $v(e_1) = 6$, $v(e_2) = 4$, $\ldots$, $v(e_6) = 7$. The accuracy added is the explicit specification of the domain and the codomain of v that allows us to distinguish it from the result of another measurement

$$W = 6, 4, 7, 4, 3, 7$$

that happens to produce the same series (but was actually the result of measuring the number of car accidents on 6 notorious crossings during one week). More importantly, the domain and codomain are essential in formulating proper preconditions for the operations that we have in mind. For instance, as we will see in the sequel, we allow two variables to be combined 'column-wise' only if they have a common domain, thus excluding the formation of data sets that make no sense because, e.g., they include the variable *number of years employed in current job* defined on a set of employees together with the variable *turnover generated* defined on a set of business transactions (we do allow such a combination though, if a relation between the employees and the transactions exists and can be made explicit through a function).

As another example, the statistic *average number of years employed in current job by income bracket* apparently returns an average for each income group that it considers. Hence it is a function $w : q \rightarrow y$ with q a set of income groups and y a set of possible average results.

Also, any model of the subject domain of a statistic (used, e.g., for statistical database modeling) can be represented as a set of functions. For instance, the knowledge that a person is a member of a household is contained in a function $z : p \rightarrow r$ with p a set of persons and r a set of households. The fact that database modeling techniques are often

based on relations instead of functions does not alter our claim: any relation can be represented by suitably chosen functions. We feel that this observation is important because it allows the (automated) permeation through the statistical process of the information contained in subject domain models.

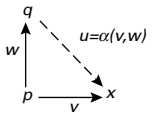
Finally, as will become clear in the sequel of the article, a statistical data set (either containing microdata or aggregated data) is properly represented as a function having a Cartesian product of value sets as a codomain. We also claim that time series data are accurately represented as a higher-order function $u : t \to y^q$ — where y^q denotes the set of functions having domain q and codomain y — that produces a data set $w : q \to y$ — i.e., a member of y^q — for each time period considered in t.

Thus, what we propose is to represent statistical data as functions, and manipulations of statistical data as operations that transform functions into functions. We indicate how this is done for aggregation.

Suppose that the variable $v : p \to x$ is *net income of a household*, i.e., p is a set of households and x is a set of possible income values. Suppose also that $w : p \to q$ assigns to each household in p its household composition in q, i.e., q is a set of household composition classes (like *single person household, or single parent household*). Then the aggregate *total net income of a household class by household composition* is a function that assigns to every composition class $d \in q$ the number

$$\sum_{\{e \in p \mid d = w(e)\}} v(e),$$

i.e., the sum of the incomes of each household in the class. It is in this sense that we perceive aggregation as an operation $\alpha(\_, \_)$ that takes two functions $v : p \to x$ and $w : p \to q$ as input, and produces a function $u : q \to x$ as output, as suggested by the diagram below.



In fact, we generalize α so that it includes other forms of aggregation as well, such as (weighted) averages. The key point (as Pursiainen 2008 notes in order to tackle the problem of consistency in aggregation) is to let x be (a set underlying) a commutative monoid: an algebraic structure with a commutative and associative operation, and an identity. To calculate a total (such as the one above), we 'feed' α with, for instance, the monoid $m = (\mathbb{R}_{\geq 0}; +, 0)$. Put differently, the monoid m acts as a parameter to α (so that we might write $\alpha = \alpha_m$) and prescribes α's mode of aggregation. Thus, to calculate a minimum, we might use $m = (\mathbb{R}_{\geq 0};\ \min, 0)$ (where $\min(n_1, n_2)$ takes the minimum of n_1 and n_2). For a product Π, we take $m = (x;\ \times, 1)$ (zap!).

Using the functional perspective outlined above, in this article we define four basic operations on statistical data, viz. functional composition, 'row-wise' and 'column-wise' combining, and aggregation (together with some 'inverse' operations, such as the 'column-wise' selection of variables). Three of them are well-known from computer

science literature (Mac Lane 1998; Barr and Wells 1999); the fourth, aggregation, is defined here in our functional perspective. Basic properties of aggregation (in the context of the other operations) are shown subsequently. For instance, we show the precise way in which the aggregation of the 'row-wise' combination of two data sets can be distributed to the aggregation of both of the data sets.

As it turns out, these properties are 'natural' in the sense that they convey formally what is expected from our informal understanding of aggregation. They should therefore come as no surprise to the reader, but that is exactly the point: we hope to convince the reader that the four simple operations are both natural and powerful to serve as the constructs of a language in which properties of statistical information can be adequately expressed. In the future, we plan to extend the language by two extra constructs: one selection mechanism intended to form subpopulations of a given population, and one mechanism to record higher-order functions such as the one mentioned earlier. However, at the moment we do not know in what way initiality can be maintained in such an extended language. More precisely: we are not sure whether properties of such a language can be expressed in the framework of equational logic.

An earlier version of this article is Gelsema (2010). There, category theory was used in order to prove the properties mentioned above: functional composition, 'row-wise' and 'column-wise' combining were inspired by category theory (as, for instance, category theory uses functional composition as its atomic operation in the category of sets and functions). However, we realized that this affected the readability of the article in a negative way. Therefore, the present article does not use category theory at all. Most of the proofs in this article are left to the reader; whenever a formal proof is required, the reader should consult Gelsema (2010).

The remainder of the article is organized as follows. In Section 1 we give a brief overview of the main objectives of the article, using a running example. Then, in Section 2, the 'column-wise' and the 'row-wise' combinations of statistical data are explained. In Section 3 aggregation is defined using commutative monoids, and properties of aggregation are stated subsequently. Then in Section 4 the implications of the framework for statistical practice are discussed. In Section 5 we compare our approach to metadata with that of two others, viz. Sundgren (1973) and Codd (1969; 1970); a proper understanding of Section 5 requires some knowledge of the two. We conclude with some general remarks on some extensions of the functional framework of statistical data and metadata presented here. Appendix A sketches the translation of the properties considered in Sections 2 and 3 to the framework of equational logic and requires some knowledge of universal algebra. From a purely scientific point of view, Appendix A constitutes the only result of the article.

This article advocates the initial algebra approach to give the exact relationship between statistical data and metadata.

1. An Example Model of a Statistical Domain

We now give an intuitive overview of the ideas presented in the article.

Consider the diagram of Figure 1 that consists of labeled boxes and labeled arrows. It is a model of a small part of the world, in the sense that it contains information about

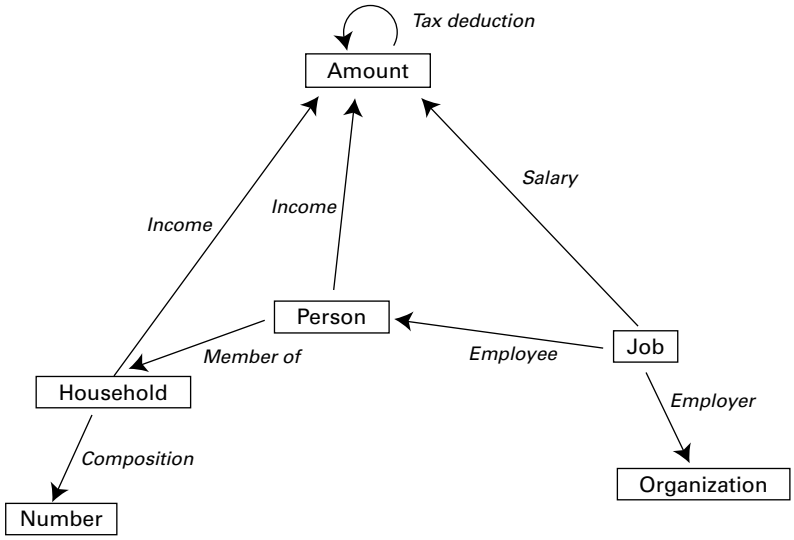


Fig. 1. An information model

phenomena that statisticians may want to measure and describe. The intuitive interpretation of the information model is straightforward and as expected: associated with a job is a person and an organization. From a job's point of view, the person associated with it is an employee, and the organization associated with it is an employer. Each person is a member of a household; the total number of persons a household contains is its household composition. The notion of an income is applicable to both persons and households. A person's income is the total amount of salary he receives (i.e., for each job he is employed in) after tax deduction. A household's income is the sum of each member's income (after tax deduction).

We can add some clarity by calling the boxes labeled *person, organization, job*, and *household* object types (Sundgren 1973), and the boxes labeled *amount* and *number* value types. Further, we can adopt the conventions that an arrow between an object type and a value type is called a variable, an arrow between two object types an object type relation, and an arrow between two (possibly identical) value types an operation. We stress that, for the time being, a proper understanding of these conventions rests upon the intuition of the reader only.

Though the information model is small, a fair amount of information that is not directly contained in it is instead implied by it. This information can be uncovered if we allow the application of some natural deductive rules.

For instance, household composition can also be treated as a variable of object type person (for the time being, this means: can be treated as an arrow that originates from the box labeled *person*) since we know that each person is a member of a household. In other words, if we ask a person about his household composition, we are dealing with the variable labeled *member of. composition*, which is the arrow we get if we follow the arrow labeled *composition* after the arrow labeled *member of*, as indicated in Figure 2. The intuitive deductive rule we apparently applied is that in a situation in which one arrow follows another, we can add their sequence without changing the information intended by

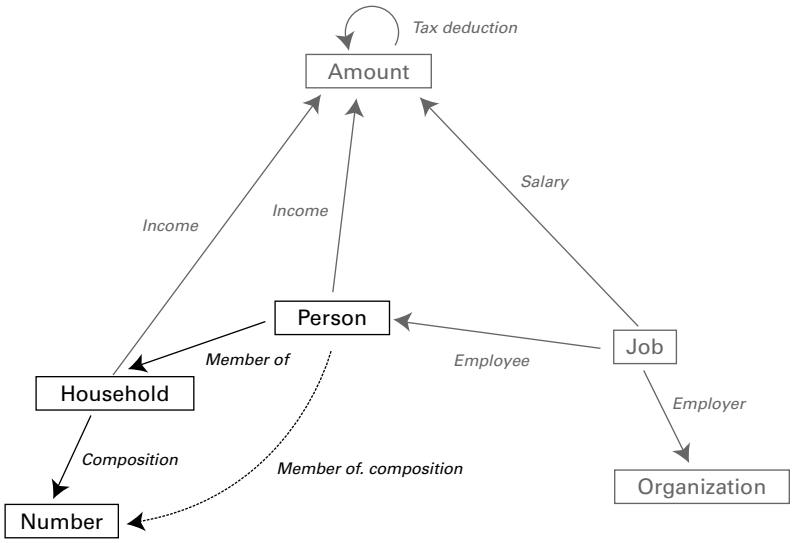


Fig. 2. Sequential composition

the model. In fact, we may choose to do this for any similar combination of arrows, for instance for a sequence of two object type relations as well (which results in an object type relation, by convention).

As another example, we may assume that any combination of two value types gives rise to a type of pairs of values. For instance, we can imagine a type denoted *number* × *amount* of pairs, the first component of any member being a number and the second being an amount. In turn, this gives rise to pairs of variables; for instance the pair denoted ⟨*composition, income*⟩ that, for each household, yields its composition and its income as a pair of values, as indicated in Figure 3. In essence, the arrow labeled ⟨*composition, income*⟩ thus may be interpreted as a microdata set consisting of two columns, the first corresponding to the variable *composition*, and the second to the variable *income*.

The informal deductive rule we applied here is that in a situation in which two arrows originate from the same box, we can add to the model a 'box of pairs' and draw to it an 'arrow of pairs' that starts from the original box. We may do the same for boxes and arrows of triples, quadruples, etc, as long as all arrows involved originate from the same box. Also, the arrows involved may be any suitable combination of variables, object type relations and operations.

The deductive rules from Figure 2 and Figure 3 immediately pose a dilemma. It is clear that, using these rules, both the arrow labeled

$$\textit{member of} \ . \ \langle \textit{composition}, \textit{income} \rangle$$

as well as the arrow labeled

$$\langle \textit{member of} \ . \ \textit{composition}, \textit{member of} \ . \ \textit{income} \rangle$$

can be deduced. The question is whether or not these arrows are synonymous, or, in other words, whether or not we may treat them as equals. To solve this question, we need to give meaning to boxes and arrows and to the deductive rules of Figure 2 and Figure 3.

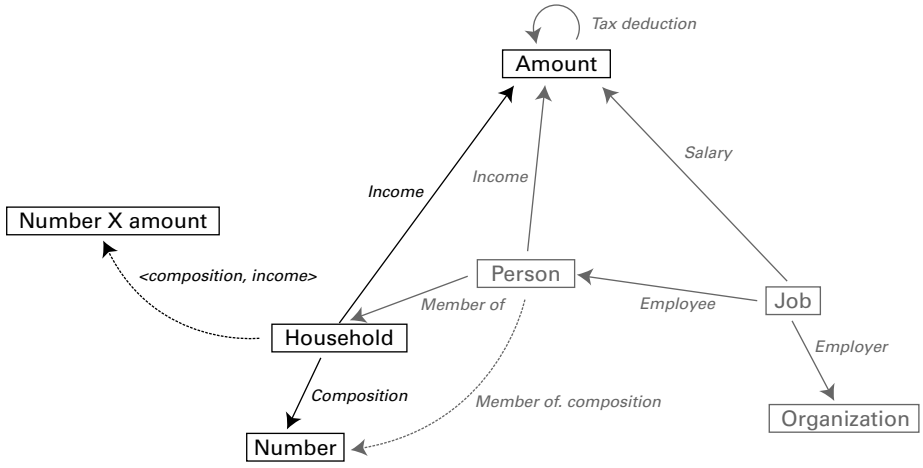


Fig. 3. Formation of a data set from two variables

We suggest that the following interpretation is both natural and sufficient: a box is an arbitrary set (for example, in the sense of Fraenkel et al. 2001) and an arrow $x \longrightarrow y$ is a function $v : x \longrightarrow y$. By a function we mean a set of pairs $\langle a, b\rangle$ such that for every a there exists exactly one element $\langle a, b\rangle$ in the set — see for instance Fraenkel et al. (2001). The sequential composition $z \cdot v$ of two arrows $v : x \longrightarrow y$ and $z : w \longrightarrow x$ is the usual functional composition: the composition $v \circ z : w \longrightarrow y$ of two functions $v : x \longrightarrow y$ and $z : w \longrightarrow x$ is defined by $(v \circ z)(a) = v(z(a))$ for every $a \in w$. The formation of a type $x \times y$ of pairs is the Cartesian product of x and y. The definition of $\langle v, u\rangle$ for functions $v : p \longrightarrow x$ and $u : p \longrightarrow y$ is given in this article (but is as expected). Using this interpretation it can be shown that the two arrows mentioned above are synonymous.

An interpretation such as the one given above, together with identities such as $\langle v, u\rangle \circ z = \langle v \circ z, u \circ z\rangle$ that hold within that interpretation, is called an algebra. There is another interpretation though, which is given by the figures themselves: this is the interpretation of simply adding suitable boxes and arrows to a given information model consisting of boxes and arrows. More precisely, arrows are interpreted as terms, such as *composition, income,* or $\langle$*composition, income*$\rangle \circ$ *member of,* and terms are considered equal, with respect to a set of equations, if they can be transformed into one another using only equations from that set. That interpretation can be considered as the initial algebra. The formal construction of an initial algebra corresponding to the information model of Figure 1 is given in the Appendix.

As a last example of a deductive rule, consider Figure 4. Now that we are equipped with a solid interpretation of arrows and boxes, we may ask ourselves whether or not the dotted arrow of Figure 4 can be deduced. In other words: which general, evidently binary, operation — say $\alpha(v, w)$ — of functions $v : x \longrightarrow y$ and $w : x \longrightarrow z$ in the specific case of Figure 4 yields the total salary of each of a person's jobs, given the salary he receives from each of his jobs individually. This article will show that such an operation α can be defined such that it is consistent with the sets-and-functions interpretation of arrows and boxes. Finally note that the arrow labeled *income* between *person* and *amount* can now be defined by the term

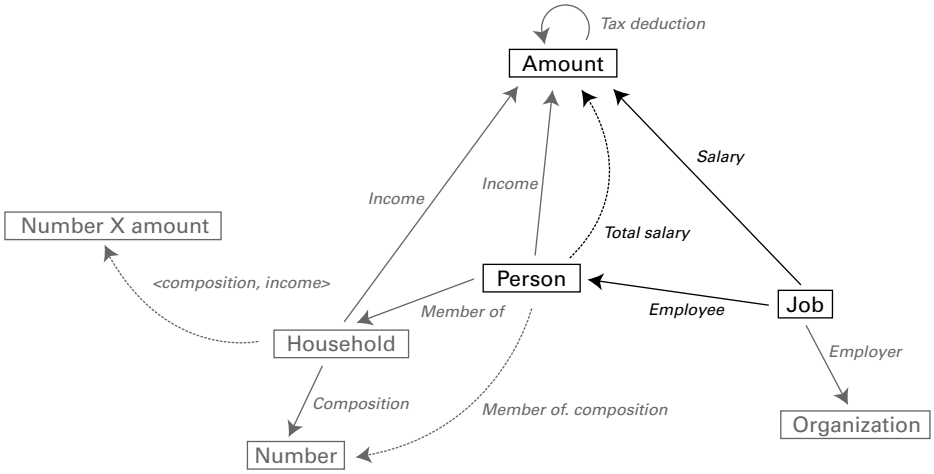


Fig. 4. Aggregation

$$income = tax\,deduction \circ \alpha(salary,\ employee),$$

which confirms in a formal way the informal definition of *income* given in the introductory part of this section. If we assume that tax is deducted using a flat tax (i.e., one tax rate for all income), then it does not matter if *tax deduction* is applied to the total salary, or to the salary of each individual job. Formally, this means that the terms below should be treated as equal:

$$tax\,deduction \circ \alpha(salary,\ employee) = \alpha(tax\,deduction \circ salary,\ employee).$$

By identifying relationships, such as the above, that hold within the sets-and-functions interpretation of statistical information, we are able to discover synonyms between terms, i.e., between descriptions of statistical data. Among a set of synonymous terms, one could be appointed as standard (or normal form), so that each statistical data item can be given a unique description. Deriving normal forms is the subject of term rewriting (Klop 1992), which is beyond the scope of this article.

2. Variables and Data Sets

The concept of a statistical variable as it is understood in official statistics — the result of a measurement of a phenomenon on a group of entities of interest — can be captured as a function $v : p \rightarrow x$. Here, p is a population, a possibly infinite set of (anonymous) entities, and x is a set of values, the possible outcomes of the measurement. The value $v(e) \in x$ is the outcome of the measurement on the entity $e \in p$. We say that a variable $v : p \rightarrow x$ is *defined on* (*the population*) p and that v is *defined for* (*the value set*) x.

We have to be careful about what we perceive as a population; in particular, we have to briefly address the role of time in measurements.

As the properties of an entity may change over time, viewing the result of a measurement as a (deterministic) function $v : p \rightarrow x$ implies that each member e of p must be fixed at

some time instance. So we assume that the members of p are in fact entities *at specific, but not necessarily identical, moments in time*. We therefore also assume that each population p comes with a map $t_p : p \to T$ that, for an entity $e \in p$, yields the moment $t_p(e)$ of its existence, and hence the time of measuring $v(e)$ (T is a universe of time instances). Though mainly of philosophical interest, we believe this is the most convenient (and accurate) approach to introduce the concept of 'time of measurement' into a general functional framework of statistical variables. Moreover, it allows us to speak of a 'population of people' and a 'population of car accidents' equally well, thus complying with the usual general definition of a population (from, e.g., Everitt 2002) that includes collections of people as well as collections of events. Note however that this philosophical assumption leads us to consider infinite populations, in particular if T is taken as infinite, so that care must be taken when defining aggregation (in Section 3).

The outcomes of two variables $v : p \to x$ and $w : p \to y$, measuring two phenomena of the same population p, can be arranged 'horizontally', i.e., in pairs, for every entity $e \in p$ through the map $e \mapsto \langle v(e), w(e)\rangle \in x \times y$, where $x \times y$ denotes the Cartesian product of x and y. The construction of this map from v and w is denoted by $\langle v, w\rangle : p \to x \times y$. The projection functions $\pi_1 : x \times y \to x$ and $\pi_2 : x \times y \to y$ defined by $\pi_1(\langle a, b\rangle) = a$ (and similarly for π_2) are associated in a natural way with the Cartesian product. In fact, they can be used to recover the variables v and w from the data set $\langle v, w\rangle$, viz. through the compositions $\pi_1 \circ \langle v, w\rangle$ and $\pi_2 \circ \langle v, w\rangle$, respectively. Thus, for suitable g, the following identity holds:

$$g = \langle \pi_1 g, \pi_2 g\rangle \tag{1}$$

where $\pi_1 g$ abbreviates $\pi_1 \circ g$ (and similarly for $\pi_2 g$).

It is easy to show that for a function $z : r \to p$ the following holds:

$$\langle vz, wz\rangle = \langle v, w\rangle \circ z \tag{2}$$

This can be shown element-wise, but is immediate from the product construction in a category (Mac Lane 1998).

We refer to the operation $\langle \_, \_ \rangle$ as the *column-wise combining (of data)*, as variables are usually listed as columns in a rectangular presentation of statistical data. To assist the intuition of the reader, the column-wise combining of two variables is sketched in Figure 5 below.

Fixing an arbitrary one-element set $1 = \{*\}$, column-wise combining of data can be extended to an arbitrary number of variables: we denote the general column-wise

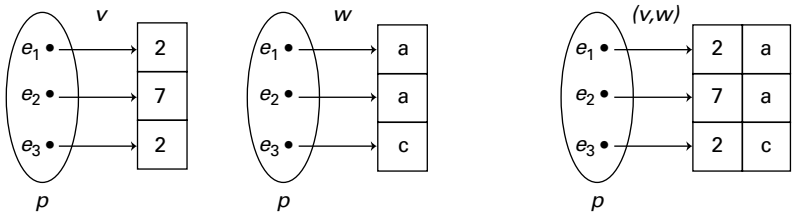


Fig. 5. Column-wise combination of two variables

combination of variables $v_i : p \rightarrow x_i$, $0 \leq i \leq k$, by $\langle v_i \rangle$, where it is understood that $\langle v_i \rangle = 1_p$ when $k = 0$, and where 1_p is the unique map $1_p : p \rightarrow 1$ (see, e.g., Pierce 2002).

Note that we thus treat variables and data sets similarly, viz. as functions.

Conversely to the situation above, we consider the *row-wise combining* (*of data*). Row-wise combining two variables v and u (or data sets for that matter) makes sense only if they have a common value set, i.e., if $v : p \rightarrow x$ and $u : q \rightarrow x$. Note that variables $v : p \rightarrow x$ and $z : q \rightarrow y$ can be forced to have a common value set $x \cup y$ by composing them with suitable injection functions $i_1 : x \rightarrow x \cup y$ and $i_2 : y \rightarrow x \cup y$. Let $p + q$ be the disjoint union of p and q, i.e., the set $p \times \{1\} \cup q \times \{2\}$, where $\cup$ denotes ordinary set union, and 1 and 2 are arbitrary but distinct elements. Note that $p + q$ coincides with $p \cup q$ (up to an isomorphism) if p and q are disjoint. Then the row-wise combination of v and u is defined as the function $[v, u] : p + q \rightarrow x$ with $[v, u](\langle e, 1 \rangle) = v(e)$ and $[v, u](\langle e, 2 \rangle) = u(e)$. So, intuitively, the map $[v, u]$ combines the outcomes of v and u 'row-wise', listing the outcomes of v for every entity in p 'on top of' the outcomes of u for every entity in q.

The variables v and u can be recovered from $[v, u]$ with the aid of the injection functions $\iota_1 : p \rightarrow p + q$ and $\iota_2 : q \rightarrow p + q$ defined by $\iota_1(a) = \langle a, 1 \rangle$ and defined similarly for ι_2. Then we have $v = [v, u] \circ \iota_1$ and $u = [v, u] \circ \iota_2$. Thus, similar to Equation 1, we obtain

$$h = [h\iota_1, h\iota_2] \tag{3}$$

for suitable h.

Also, it is easy to show that, opposite but similar to Equation 2, we have

$$[zv, zu] = z \circ [v, u] \tag{4}$$

for every function $z : x \rightarrow y$. This is immediate from the coproduct construction in a category (Mac Lane 1998), but can also be shown element-wise, if preferred.

We refer to the operation [_, _] as the *row-wise combining* (*of data*). As with the column-wise combining of data, row-wise combining of data is easily extended to any number of variables: we denote the general row-wise combination of variables $u_i : p_i \rightarrow x$, $1 \leq i \leq k$, by $[u_i]$, where it is understood that $[u_i] = 0_x$ when $k = 0$, and where $0_x : \emptyset \rightarrow x$ is the unique map with codomain x (see Pierce 2002). The row-wise combination of two variables is sketched in Figure 6 below.

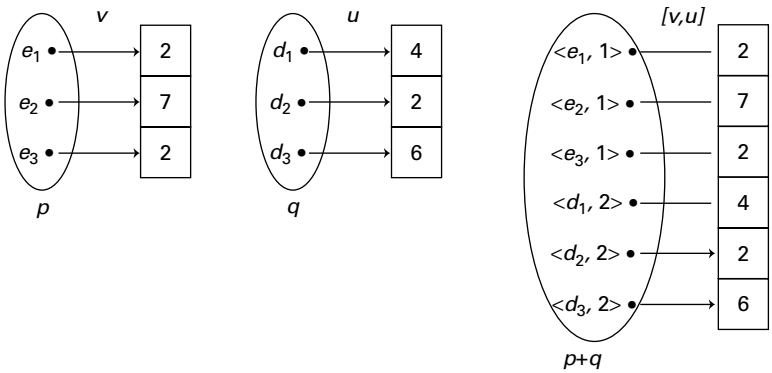


Fig. 6. Row-wise combination of two variables

The operations considered in this section are the basic operations for constructing data sets from statistical variables, both column-wise and row-wise.

3. Properties of Aggregation

In this section we define in a general way the notion of aggregation. As suggested in the Introduction, aggregation is an operation $\alpha(v, w)$ defined on two variables $v : p \rightarrow x$ and $w : p \rightarrow q$, that outputs a function $u : q \rightarrow x$, as indicated in Figure 7 below. The operation α can be seen as a formal confirmation of the idea that, for instance, the aggregate *total turnover of Dutch enterprises by activity* should somehow depend on two variables (defined on the same population), viz., *turnover of a Dutch enterprise* and *activity of a Dutch enterprise*.

As stated in the Introduction, our notion of aggregation is based on commutative monoids; we briefly review the definitions involved (see also, e.g., Grillet 2001).

A monoid m is a triple $(m; \cdot, 1)$ where m is a set, $\cdot$ is an associative binary operation on m, and 1 is an element of m that is an identity for $\cdot$, i.e., for all $a, b, c \in m$ we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ and $a \cdot 1 = a = 1 \cdot a$. Note that we identify a monoid m with its underlying set m; it will always be clear from the context whichever we mean. A homomorphism between two monoids m and n is a function $h : m \rightarrow n$ such that $h(a \cdot b) = h(a) \cdot h(b)$ for every $a, b \in m$, and $h(1) = 1$. A monoid is commutative if $a \cdot b = b \cdot a$ for all $a, b \in m$. Since all monoids we consider here are commutative, we usually refer to them as *monoid* from here on (as it is also understood that by homomorphism we mean monoid homomorphism). In fact, for a commutative monoid m, it is usual to employ an additive notation: $m = (m; +, 0)$, which we will follow from here on.

The product of two monoids m and n is the monoid $m \times n$ (with underlying set $m \times n$, i.e., the Cartesian product of m and n), with $\langle a_1, b_1 \rangle + \langle a_2, b_2 \rangle = \langle a_1 + a_2, b_1 + b_2 \rangle$, and with identity $\langle 0, 0 \rangle$. The product of monoids is easily extended to an arbitrary number of monoids.

Let m be a monoid, $v : p \rightarrow m$ a variable, and let $p' = \{e_1, \ldots, e_k\}$ be a finite subset of p. The *generalized sum of v by p'*, denoted

$$\sum_{e \in p'} v(e),$$

is defined as $v(e_1) + \cdots + v(e_k)$ if p' is nonempty, and where

$$\sum_{e \in p'} v(e) = 0$$

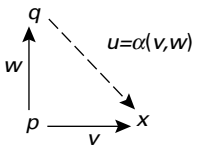


Fig. 7. The aggregation of v by w.

if $p' = \emptyset$. Note that this definition makes sense, since it does not depend on the order of the elements in p', by associativity and commutativity of $+$.

A function $w : p \rightarrow q$ is *inverse-finite* if $w^{-1}(d) = \{e \in p \mid d = w(e)\}$ is finite for every $d \in q$. Note that w is inverse-finite if p is finite. Also, if w is inverse-finite, then p is finite if q is finite. Every injection is inverse-finite. The composition of two inverse-finite functions is inverse-finite. If $w_1 : p_1 \rightarrow q$ and $w_2 : p_2 \rightarrow q$ are inverse-finite, then so is $[w_1, w_2]$.

Our interest in inverse-finite functions for the purpose of aggregation is the following. We want to find a general expression for the operation that takes a variable $v : p \rightarrow m$ that has a binary commutative operation $+$ defined on its value set m, together with a second variable $w : p \rightarrow q$, and that produces a variable $u : q \rightarrow m$ defined as the map

$$d \mapsto \sum_{d=w(e)} v(e) \tag{5}$$

for every $d \in q$. For this to make sense it is a requirement that the sets $w^{-1}(d)$ are finite, for otherwise the generalized sum would fail. So, let $w : p \rightarrow q$ be inverse-finite, and v, p, and m as before. The *aggregation of v by w*, denoted $\alpha(v, w)$, is the map $u : q \rightarrow m$ defined by (5). Note that (5) is well-defined, as its notation must be seen as an abbreviation of

$$d \mapsto \sum_{e \in w^{-1}(d)} v(e)$$

which corresponds to the notation in the definition of the generalized sum.

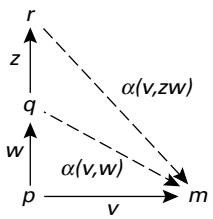
The following two properties of α are immediate: let v, w, p, q, and m be as before, let $z : q \rightarrow r$ be inverse-finite and let $h : m \rightarrow n$ be a homomorphism. Then we have

$$\alpha(v, zw) = \alpha(\alpha(v, w), z) \tag{6}$$

and

$$\alpha(hv, w) = h \circ \alpha(v, w) \tag{7}$$

which show the distribution of α with respect to the functional composition in both of its arguments. The situation of Property (6) is clarified by the diagram below.



It is confirmed by the fact that, for all $c \in r$, we have

$$\sum_{c=z(w(e))} v(e) = \sum_{c=z(d)} u(d),$$

with $u : q \rightarrow m$ the Map (5) above. The proof of this fact is straightforward, but a bit tedious, and is left to the reader. Property (7) is just a generalized homomorphism condition for h, as it expresses that

$$h(v(e_1)) + \cdots + h(v(e_k)) = h(v(e_1) + \cdots + v(e_k)).$$

The reader requiring a more algebraic proof of Properties (6) and (7) is referred to Gelsema (2010).

Property (6) is explained intuitively as follows. Suppose that we have access to the following information: for every entity e in a set p of people, we know the household $w(e) \in q$ he or she is a member of, as well as his or her income $v(e) \in m$ (with $m = (\mathbb{N}; +, 0)$ say). Also, each household d in q is assigned a class $z(d) \in r$, based, say, on the region in which it resides. Then the aggregate *total income earned per region* can be computed in one stroke, adding together the incomes of the persons that reside in the same region. For this we must use zw : the map that assigns to a person the regional class of his household. It can also be computed in two strokes, by first calculating the income of each household, adding together the incomes of its members, and then by calculating the total income earned per region, adding together the incomes of the households in the same region. Of course, the two results must be equal.

Suppose that, in addition, we consider income taxes in calculating the total income of a household. For simplicity, we assume a flat income tax, i.e., one tax rate is applied to all income. Then it does not matter whether tax is deducted from the income of each member of a household after which the results are added, or conversely, incomes are added after which tax is deducted from the total. In other words, tax deduction is an endomorphism on $m = (\mathbb{N}; +, 0)$ (i.e., a homomorphism between m and itself). This shows the intuitive meaning of Property (7).

It makes sense to view $\alpha(v, w)$ as the functional composition $\gamma(v)\ \delta(w)$ of $\gamma(v)$, called the *elementary class parameter induced by v*, and $\delta(w)$, called the *dimensional structure induced by w*. Separating α in this way formally establishes the idea that the aggregates *total turnover of Dutch enterprises by activity* and *total turnover of Dutch enterprises by size class* share a class parameter, but differ in their dimensional structure. Thus, the notions of a class parameter and a dimensional structure, at least from an information-sharing point of view, seem more fundamental than the notion of an aggregate $\alpha(v,\ w)$. Speaking in terms of the pharmacist's cabinet analogy of the Introduction, separate drawers will be put aside for $\gamma(v)$ and $\delta(w)$. They are defined in a straightforward way as follows.

For a set p, let Fp be the set of finite subsets of p and let v be as before. Then we define $\gamma(v) : Fp \rightarrow m$ simply as the generalized sum of v, i.e., $\gamma(v)$ maps $p' \in Fp$ to $\sum_{e \in p'} v(e)$. We warn the reader that $\gamma(v)$ is not a homomorphism between m and the monoid we get from Fp by taking set union as operation and the empty set as identity. Now let w be as before. Then the map $\delta(w) : q \rightarrow Fp$ is simply w^{-1}, i.e., $\delta(w)$ maps an element $d \in q$ to the set $\{e \in p \mid d = w(e)\}$. Note that it makes sense to compose $\gamma(v)$ with $\delta(w)$ as their composition 'factors through' Fp. It is trivial to verify that viewing $\alpha(v,\ w)$ as $\gamma(v)\ \delta(w)$ does not alter Definition (5), nor does it affect Properties (6) and (7).

To see that aggregation is not limited to summation, but also includes, e.g., calculating weighted averages, consider the following example.

Example 3.1 (Weighted arithmetic means, adapted from Pursiainen (2008)) Let x and y be the sets $\mathbb{R}$ and $\mathbb{R}_{>0}$, respectively. Suppose that the first is used as a value set of

a variable $v : p \rightarrow x$ (p is an arbitrary population), and the second, y, is used as a set of positive weights for the values of v, which we record as a function $z : p \rightarrow y$. The interpretation of z is the following: for each value $v(e) \in x$ the corresponding weight that should be used in calculating weighted averages of v is $z(e) \in y$. Let $w : p \rightarrow q$ be any inverse-finite function, i.e., q is interpreted as a set of 'categories' d which are used to denominate 'classes' $w^{-1}(d)$ of interest within the population p. Now let m be the monoid defined by taking the underlying set $m = x \times y \cup \{\omega\}$, together with the following binary operation (we denote ordered pairs $\langle a, b\rangle$ by (a, b) here):

$$(a_1, b_1) \cdot (a_2, b_2) = \left(\frac{a_1 b_1 + a_2 b_2}{b_1 + b_2}, b_1 + b_2 \right),$$

and such that $e \cdot \omega = e = \omega \cdot e$ for every $e \in m$ (so ω is m's identity). The first component gives the weighted arithmetic mean of a_1 and a_2, and the second adds their corresponding weights b_1 and b_2. It is easy to check that $\cdot$ is associative and commutative. Now the function $\alpha(\langle v, z\rangle, w) : q \rightarrow m$ returns a pair consisting of the weighted average of v together with the sum of the weights used for each category in q. □

Next, we study the aggregation of data sets that are composed row-wise or column-wise, as defined in the previous section. We consider binary combinations only, as they are easily generalized to arbitrary combinations.

Let $v_1 : p \rightarrow m_1$ and $v_2 : p \rightarrow m_2$ be two variables with m_1 and m_2 monoids, and let $w : p \rightarrow q$ be inverse-finite. Then we have

$$\alpha(\langle v_1, v_2\rangle, w) = \langle \alpha(v_1, w), \alpha(v_2, w)\rangle \tag{8}$$

Note that in the left-hand side of Equation (8), aggregation is meant with respect to the monoid $m_1 \times m_2$.

To explain Property (8), let v_1 be the variable v as before, i.e., v_1 measures a person's income. Let also w be as before, i.e., w assigns a household to the respective person. Let $m_2 = m_1 = (\mathbb{N}; +, 0)$ and let v_2 be the constant 1 everywhere. Then $\alpha(v_2, w)$ is household composition, i.e., the number of persons a household contains. Now in order to produce a data set containing the total incomes of a set of households as well as their compositions, we may first form a data set of persons containing their income together with a column of 1s (i.e., the variable v_2) and then sum both columns at the same time for each household. We may also calculate household composition and household income separately, and then combine their results column-wise; of course the outcomes of both procedures should be identical.

Property (8) uses the fact that, for an arbitrary finite subset p' of p, we have

$$\sum_{e \in p'} \langle v_1(e), v_2(e)\rangle = \langle \sum_{e \in p'} v_1(e), \sum_{e \in p'} v_2(e)\rangle,$$

which is immediate from the definitions of the generalized sum and the product of two monoids.

To set up the properties of the aggregation of a row-wise combined data set, let $v_1 : p_1 \to m$, $v_2 : p_2 \to m$, and let $w_1 : p_1 \to q$ and $w_2 : p_2 \to q$ be inverse-finite. Then

$$\alpha([v_1, v_2], [w_1, w_2]) = \alpha(v_1, w_1) + \alpha(v_2, w_2), \tag{9}$$

where it is understood that, for functions $u_1, u_2 : q \to m$, by $u_1 + u_2$ we mean the monoid operation $+$ applied to the data set $\langle u_1, u_2 \rangle$, or, in other words,

$$u_1 + u_2 = (+) \circ \langle u_1, u_2 \rangle.$$

Though intuitively simple to understand, the proof of Property (9) is a bit more complex. The reader may try it him- or herself, or may consult Gelsema (2010) for a proof using category theory.

Intuitively, Property (9) considers a population p of, say, people, that is split into two, say males p_1 and females p_2. Then the calculation of a household's total income can be distributed to calculating its males' share and its females' share. What then remains to be done is to add their results for each household.

Finally, picturing aggregation as in Figure 7, it is natural to consider those cases for which the diagram commutes, i.e., for which $\alpha(v, w) \circ w = v$ holds. The cases considered below are among those.

Let $0_{p,m} : p \to m$ be the map $e \mapsto 0$, for all $e \in p$ (where 0 is the identity of m) and let $w : p \to q$ be inverse-finite. Then

$$\alpha(0_{p,m}, w) = 0_{q,m}. \tag{10}$$

Let $v : p \to m$ and let 1_p be the identity on p. Then

$$\alpha(v, 1_p) = v. \tag{11}$$

Let v be as above and let w be an injection. Then

$$\alpha(v, w) \circ z = v. \tag{12}$$

These properties are all easy to show (and most are left to the reader, or see Gelsema 2010). For instance, using (12) to show (11), we have $\alpha(v, 1_p) = \alpha(v, 1_p) \circ 1_p = v$.

The cases above are not exhaustive — in fact we conjecture that if v and w are functions for which it holds that

$$\text{for all } e \in p,\ w \text{ is injective on } e \text{ or } v(e) = 0,$$

then we also have $\alpha(v, w) \circ z = w$. The reader can verify that even this condition does not cover all cases.

Example 3.2 Consider the statistical domain of Figure 8, which is an extended version of the information model of Figure 1. The dotted arrows of Figure 8 represent derived notions. For instance, the income of a household, labeled *hh income* in Figure 8, is now formalized as α(*income, member of*), i.e., the sum of every member's income. Also, household *composition,* the number of persons a household consists of, is now defined as α(1, *member of*), where the arrow 1 designates the map from *person* to *number* that is 1 everywhere. The box labeled *avg. number* designates the monoid of Example 3.1 and the arrow labeled *i* is the inclusion of the product *number* × *number* in it. The arrows labeled π_1 and π_2 are the projection arrows defined in Section 2.

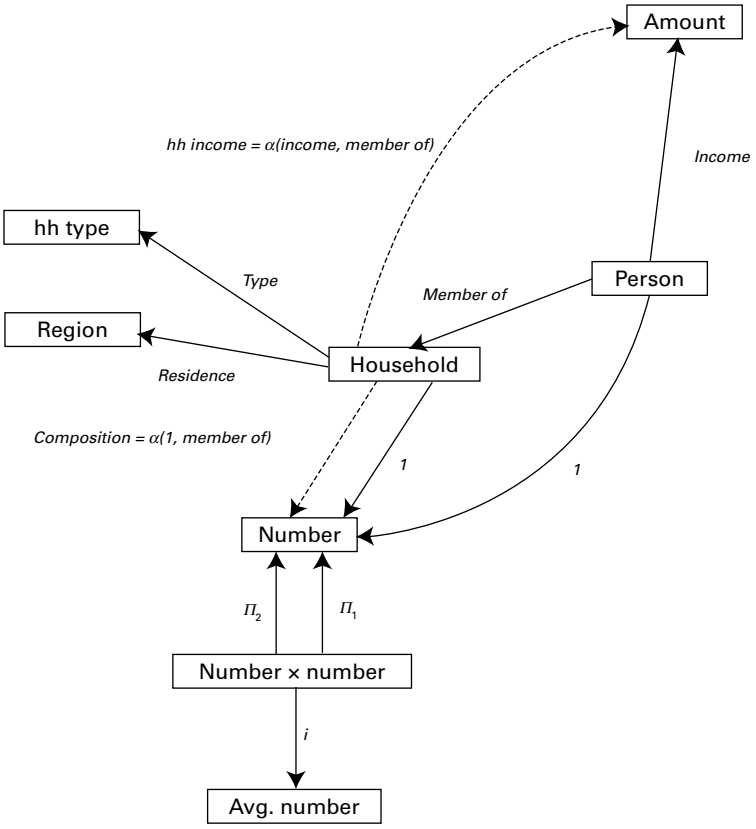


Fig. 8. An example of a statistical domain

Associated with each household is now a *type* (such as *single parent household* or *couple with children* for instance) and the region of its *residence.*

Suppose the aim is to give an metadata term in normal form for the data set informally typified by *average composition and total income of households by type and residence.* First, it should be clear now that *total income of households by type and residence* is given by the term α(*hh income,* ⟨*type, residence*⟩). Second, using the monoid of Example 3.1 with weight vector 1 (depicted by the arrow labeled 1 from *household* to *number*), the *average composition of households by type and residence* is $\alpha(i \circ \langle composition,\ 1\rangle, \langle type,\ residence\rangle)$. Column-wise combining the two gives

$$N_1 = \alpha(\langle i \circ \langle composition, 1\rangle,\ hh\,income\rangle, \langle type,\ residence\rangle)$$

by Property 8. If the general strategy is to rewrite expressions using Property 8 in the right-to-left direction, then the expression N_1 above is in normal form: it cannot be rewritten any further. If however the rewrite strategy is to apply Property 6 in the right-to-left direction (and the other properties in either direction), then the following equivalent expression is in normal form

$$N_2 = \langle \alpha(i \circ \langle composition, 1\rangle, \langle type, residence\rangle),$$

$$\alpha(income, \langle type, residence\rangle \circ member\ of\ )\rangle$$

as the reader can verify.

Finally, almost trivially now, if (with respect to a global homomorphism h) the terms *income, member of, type* and *residence* correspond to the variables *v, w, z,* and *u,* respectively (i.e., $h(income) = v$, $h(member\ of) = w$, etc.), then the expression

$$h(N_1) = \alpha(\langle i \circ \langle h(composition), 1\rangle,\ h(hh\ income)\rangle,\ \langle h(type),\ h(residence)\rangle)$$

$$= \alpha(\langle i \circ \langle \alpha(1, w), 1\rangle,\ \alpha(v, w)\rangle, \langle z, u\rangle)$$

calculates the actual data set appointed by N_1 (or N_2 for that matter). □

The notion of aggregation that is considered in this section does not take into account the fact that an output table produced in a statistical office often contains statistical data for different levels of detail. For instance, it is common that an output table like *total turnover of Dutch enterprises by activity* records the turnover of enterprises for activity categories like, e.g., *construction* and *manufacturing,* as well as for 'smaller' categories like *civil engineering* and *manufacture of textiles.* The relationships that hold between figures recorded for different levels of detail ('smaller' categories should add up to 'larger' categories) is something that cannot be captured using the notion of aggregation considered in this section. However, in Gelsema (2010) an extension of the framework of this section is considered: using Boolean algebras for structuring classification systems, a notion of aggregation is described that respects the relationships mentioned above.

By identifying the equations that hold between the different operators considered, this section and the previous sketched the intended semantics for our functional view of statistical information. We stress that this is almost all we need to gain the benefits explained in the Introduction, in particular the homomorphism h between metadata terms and data items. The only thing that remains to be done is to translate the equations into the framework of equational logic; once we have done that, the existence of the initial algebra follows from Meinke and Tucker (1992). This translation is sketched in Appendix A.

4. Discussion

The main arguments presented in this article are twofold. First, statistical information elements of different kinds are best represented by functions. Whether for a variable, a microdata set, an aggregate, an object type relation, or a statistical operation: a function captures its true meaning. Second, the initial algebra is a way to model statistical metadata in a faithful way. Using a modest set of operations, a simple algebra of functions captures the dependencies between the information elements listed above, such as the relationship between a data set and its variables. The terms of the associated initial algebra are the 'best' descriptors of such information elements, since they provide a natural and compositional semantics. This means, for instance, that a data set is described using the descriptions of its variables, which is a natural requirement.

The objective of the article is to present a formal framework for statistical metadata. It is felt that such a framework can contribute to statistical practice in a number of ways. These are listed below using the framework's main virtues.

First, we have given much attention to the formality of the framework. This has been done in order to give all kinds of automated processes a sound basis for capturing the information objects they operate upon. In other words, supporting automation is the main objective of the formality of the framework. The processes that we have in mind are the ones that concern the design of statistical output and intermediate results, in such a way that interdependencies between them are maintained across the statistical process (these processes reside in the column "Design" of the Generic Statistical Business Process Model (GSBPM); see Vale 2009). In this way, for instance, the effect of changes to the input on the output of a statistical process can be made visible. Also, it is felt that the algebraic approach of the framework supports the automated generation of metadata during statistical production. Both uses are examples of the "industrial" view of statistical production, proclaimed, for instance, by van der Veen (2011).

Second, the framework is formulated in a precise way. Definite choices have been made in defining the concepts of a statistical variable, a statistical data set, etc. Therefore we believe that the framework could play a role in settling discussions about the semantics of statistical information objects, as in our experience those discussions are often central (and can be lengthy without the proper means) when designing statistical information models. The recent initiative of the Generic Statistical Information Model (GSIM; see van der Veen 2011), the counterpart of the GSBPM, specifically includes a semantics part and we feel our framework could make a valuable contribution to that, in the sense meant above.

Third, we have tried to keep the framework as simple as possible, concentrating on the main issues. The algebraic approach makes it relatively easy to extend the framework to new functionalities, viz. by adding operations and equations. Existing standards, such as SDMX, have received some criticism recently on their complexity (see, e.g., UNECE Secretariat 2011) calling for "light" versions of those standards. This complexity is highlighted by the decisions a designer of SDMX artefacts must make: often there are numerous different (but 'equivalent' in a sense left obscure) ways possible in the design of such artefacts, counteracting their comparability and counteracting standardization. Again, it is believed that simpler (exchange) standards can be developed, provided that they are based on frameworks such as the one presented in this article. Also, we believe that every engineering discipline should be guided by some sort of theory, and the designing of information models should be no exception to that. The first steps towards such a theory are the ones presented in this article.

5. Comparison with Other Approaches

The goals set by this article probably resemble those of Sundgren (1973) most: both seek to discover the structure of statistical information. Our technical approach however is more similar to Codd (1969; 1970). We draw a comparison with each of those works in this section.

Most of the constructs that we treat in this article (viz. composition, product, sum, and aggregation), Sundgren also acknowledges in some form or the other. To begin with, the notion of a variable, that Sundgren defines as a single-valued attribute, probably

corresponds to our use of a function, provided we may treat the "relevance group" of that attribute as the domain of the function. The motivation for this is reinforced by the construct of "consolidated constellation", a grouping construct of attributes, which then corresponds to our product construction, since consolidated constellation is only meaningful for attributes that have identical relevance groups. Also, it is believed that the "generation of relation dependent properties", a particular closure rule of properties of objects, can be explained by our functional composition. Further, the notion of aggregation that Sundgren uses seems to suggests a monoid or semigroup. Finally, the "box algebra" Sundgren uses to explain the dimensional structure of aggregates bears much resemblance to a Boolean algebra, and can presumably be exchanged for it.

The approach of Sundgren considers many other relevant notions though. We believe that many of those can be understood by proper refinements of our functional view. For instance, the useful notion of an "identifying property" can be understood by an injective function: a variable $v : p \rightarrow x$ is *identifying* if for each moment m in time T the restriction of v to p_m is an injection where $p_m = t_p^{-1}(m) = \{e \in p | t_p(e) = m\}$.

Though technically profound, Sundgren (1973) chooses a nonstandard approach in defining its constructs: it does not consistently use a well-understood underlying theory (such as category theory or algebra) upon which its constructs can be grounded. The lack of an underlying theory is the main difference between our approach and that of Sundgren; without it, we feel that the identities of Sections 2 and 3 cannot be derived, and indeed Sundgren does not try to discover them. Also, Sundgren does not seem to be interested in keeping the number of its basic constructs limited. In contrast, we believe that the value of our approach lies in keeping the number of language constructs small, yet retaining a language expressive enough to explain, for instance, the great number of seemingly independent notions Sundgren considers. Thus, we feel that our approach is more fundamental.

Further, Sundgren does not treat the relationships between data and metadata in a satisfactory way, viz., by considering some kind of (formal) mapping between the two, as we do. Finally, we believe that a lack of an underlying theory may also lead to some undesired results. As an example, because in an $\alpha\beta\gamma$-query the relevance group of an aggregated variable (the β-part) depends on the boxes given by the variables in the γ-part, technically a query like *total income by sex* not only differs in its γ-part from *total income by social class,* but necessarily also differs in its β-part, which is unsatisfactory and probably not what was intended. We resolve this technical issue by letting our aggregates 'factor through' Fp, the set of finite subsets of (a population) p, or, informally put: the relevance group of *total income* is the set of finite subsets of the relevance group of *income.* This makes *total income* a separate notion, i.e., one that is independent from *social class* or *sex.*

We believe that the use of some common underlying theory is indispensable for truly giving meaning to the nature of statistical information. Because of the agreement of ideas however, we think of our approach as a formal semantics for many of the notions of Sundgren (1973).

As for Codd (1969; 1970), and many initiatives based on the relational calculus since then (see, e.g., Elmasri and Navathe 1989), their goal is to describe a small set of elementary relational operations that can form the basis of a query language. The objective of Codd is to also study their dependencies (composition is defined through join for

Table 1. A three-column table of statistical data

Size class	Activity	Turnover
10	Agriculture	12
10	Agriculture	24
10	Industry	72

instance, and join is defined through projection) which classifies this approach as algebraic in nature.

Indeed, Codd (1970) considers (relational) composition as an (elementary) operation, as we do. Further, our product construct can be seen as a restricted kind of join. Only the restriction operator of Codd (1970) is lacking in our approach (however, see the Conclusion of this article). On the other hand, aggregation is not mentioned in Codd (1969; 1970) and the author knows of no other initiative to supplement the relational calculus with aggregation in a satisfying way. This is despite the fact that most commercial database management systems are equipped with aggregation operators nowadays.

However, we strongly argue against a relational view of statistical data, and propose our functional view instead. To begin with, any relation can be expressed (without any loss of information) as a set of functions. For instance, a binary relation R on domains A and B (i.e., $R \subseteq A \times B$) is given by two functions $l : R \rightarrow A$ and $r : R \rightarrow B$ defined by $R = \{\langle l(o), r(o)\rangle \mid o \in R\}$, and similarly for an arbitrary n-ary relation. Thus, in this sense, the relational approach is not 'more expressive' than the functional. Secondly, a sequence of observations such as 4, 7, 4, 3, 7, while suitable for statistical purposes as is, in the relational view needs an additional attribute that (possibly combined with the sequence) can serve as a key. Only this will prevent it from reducing to the set {4,7,3} which, according to Codd (1969; 1970) is what is left of the sequence when viewed as a relation. Probably for this reason, contrary to Codd, today's relational database management systems operate on multisets (Engelfriet and Gelsema 1999) of records, instead of sets of records. Thirdly, using relations (and operations on them) for the representation of statistical information can lead to nonsense, as the following example shows. Consider Table 1, which for three enterprises reports their size class (according, e.g., to the number of people employed), their main activity (using some classification of activities, e.g., NACE), and their turnover (in millions). Note that any of its columns can be deleted just like that (the deletion of a column of data is accomplished by the projection operator of Codd (1969; 1970)). For instance, when the second column labeled *activity* is deleted, what is left is a table that is just a little less informative than the one started with, but nevertheless useful for statistical purposes: the *average turnover by size class* for instance can still be computed from it. Now consider Table 2 that results from Table 1 by computing the *average turnover by size class and activity*. From a relational perspective the second table is similar to the first, but its statistical intention is entirely different.

Table 2. Average turnover by size class and activity

Size class	Activity	Avg. turnover
10	Agriculture	18
10	Industry	72

Table 3. After deletion of the middle column

Size class	Avg. turnover
10	18
10	72

This can be seen by again deleting the second column, admissible from a relational point of view, which results in Table 3. From a statistical point of view though, that table has become useless, since its dimensional structure has been corrupted. Note also that taking the average of the two figures in Table 3 would not lead to the correct *average turnover by size class*, according to Table 1.

Our functional perspective of statistical data solves this issue by viewing the first table as a function v : *enterprise* $\rightarrow$ *size class* $\times$ *activity* $\times$ *turnover* but the second as a function w : *size class* $\times$ *activity* $\rightarrow$ *avg.turnover* and by the restriction that our projection operator is defined only for 'columns on the right side of $\rightarrow$ ', cf. Section 2.

6. Conclusion

We have presented a general framework for elementary operations on statistical data, as they are understood in official statistics. If these operations are seen as operations in an algebra, then the corresponding term algebra is a natural candidate for the structure (or: the syntax, or: the format) of the metadata that describe statistical data.

In official statistics, the lack of a theory of statistical information (with the exception of Sundgren (1973)) is prominent in the proliferation of metadata models, each claiming to have found the 'right' interpretation of "data about data" but, in effect, each failing to make precise the relationship between a metadata item and the corresponding data item. In addition to the results recited above, this article advocates being precise about statistical data and metadata in such a way that meaningful and verifiable claims about metadata models can be formulated. Further, it attempts to offer the early steps towards a general theory of information in official statistics. Again, we feel that the availability of such a theory is crucial; the most prominent reason being that — as we claim — very few of the metadata models used in statistical offices are sufficiently and satisfiably closed under aggregation, by which we mean that either the result of aggregating microinformation falls out of the model's scope, or the model treats aggregated information as if it were microinformation (or vice versa). Another rationale is that such a theory is likely to be of use in the harmonization and the increase of coherence within official statistics.

The next step towards such a theory is to properly account for time series information and related phenomena. For instance, ideally, variables such as *output of the production of shoes, output of the production of cars*, etc., are treated as particularizations by product group of one variable, viz. *output*. The proper management of these related variables, we believe, is solved by exponentiation. The expressiveness of the result will be equivalent to that of a Cartesian closed category (Mac Lane 1998) enriched with the notion(s) of aggregation developed here (and together with the notion of classification developed in Gelsema (2010)). The final extension is a mechanism for constructing a subpopulation from a given population. This mechanism should acknowledge the fact that the definition

of a subpopulation of entities is necessarily founded on similarities in the outcomes of one or more variables defined on the given population (after all, this is the only means within a statistical office to effectively decide whether or not an entity is a member of a subpopulation). We believe that the notion of a subobject from category theory offers the right approach. However, it may turn out that the resulting equations cannot be expressed within the framework of equational logic.

Appendix A. Sketch of an Initial Algebra

This section is of a technical nature and may be skipped by the reader unfamiliar with universal algebra. It is meant to support the claim that an initial algebra exists for the type of algebras described in the Introduction.

We sketch the construction of an initial algebra for the information model of Section 1 using the identities of Sections 2 and 3. For brevity and simplicity, we will consider binary products only and we leave out binary sums (and thus row-wise combining) altogether. For the same reason we will not consider the identities (10), (11) and (12) of Section 3 that concern the commutativity of Diagram 7. Finally, we will not treat $\alpha(v, w)$ as the derived operator $\gamma(v)\delta(w)$. We stress however that these simplifications do not affect the main claim of this section.

The algebraic framework we use is that of equational logic: it is known that algebraic classes described in this framework admit initial algebras (Meinke and Tucker 1992). More precisely, for a set of sorts S, and an S-sorted signature Σ, to be defined below, we give an equational axiomatization E (i.e., a set of equations) that corresponds to the set of equalities identified in the previous sections. Then the class of all models of E, i.e., the class of algebras for which every equation in E is valid, contains an initial algebra.

We start by defining the set S of sorts. Let O' be the set of object types from Figure 1, and let M' be the set of value types. So O' consists of the object types *person, organization, job,* and *household*, which we abbreviate with *p, o, j*, and *h*, respectively. The set M' contains *amount* and *number*, which we will denote by *a* and *n* respectively. Let P' be the union of O' and M'. We construct the set P of product types inductively as follows:

$$P_0 = P'$$
$$P_i = \{(p_1 \times p_2) \mid p_1, p_2 \in p_{i-1}\}, \text{ where } i > 0, \text{ and}$$
$$P = \cup_{i \geq 0} P_i.$$

We let the subset M of P consist of those product types that are inductively built up from M' (i.e., by taking $M_0 = M'$ and defining M similarly to P above). The set S of sorts is now $S_1 \cup S_2 \cup S_3$ where

$$S_1 = \{(p \rightarrow q) | p, q \in P\},$$
$$S_2 = \{(p \rightharpoonup q) | p, q \in P\}, \text{ and}$$
$$S_3 = \{(m \hookrightarrow n) | m, n \in M\}.$$

The sets S_1, S_2, and S_3 contain the sorts for functions, inverse-finite functions, and monoid homomorphisms, respectively.

We now construct an S-sorted signature Σ for the algebra we have in mind. As is usual, Σ is an indexed family of sets $\Sigma_{w,s}$ with $w \in S^*$ and $s \in S$, that contain those operation symbols that expect arguments of sorts $w = s_1 \cdots s_l$ (with $s_i \in S$) respectively, and that return sort s. As special cases, $\Sigma_{\lambda,s}$ contain the constant symbols of sort s, which we will define first.

One part of the constant symbols of our algebra simply consist of the arrows of Figure 1, except for the two arrows labeled *income* and the arrow labeled *composition* (this is because we will derive them from the other arrows, using the operations considered below). Taking $s = (j \rightarrow a)$ for instance, then $\Sigma_{\lambda,s} = \{salary\}$, i.e., the arrow labeled *salary* between the object type *job* and the value type *amount*. The arrows *member of, employee*, and *employer* represent inverse-finite functions; note that this is in accordance with the philosophical discussion of Section 2. The arrow *tax deduction* represents the only homomorphism of Figure 1. Thus, for instance, we let $\Sigma_{\lambda,(j \rightarrow o)} = \{employer\}$ and we let $\Sigma_{\lambda,(a \hookrightarrow a)} = \{tax\,deduction\}$.

Another part of the constant symbols represent the monoid operations. For each $m \in M$, we let $\Sigma_{\lambda,((m \times m) \rightarrow m)} = \{\mu_m\}$ where μ_m is the symbol that represents the monoid operation of m. For the value types a and n, we plan to implement those by addition (but that happens in a later stage; for now μ_m is nothing more than an S-sorted constant symbol).

Finally, we treat the projection functions π_1 and π_2 as constants: for every $p = (p_1 \times p_2) \in P$, we let $\pi_{1,p} \in \Sigma_{\lambda,(p \rightarrow p_1)}$ and similarly for $\pi_{2,p}$.

We now give signatures to the operations $\circ$, $\langle \_, \_ \rangle$ and α. We use a trick to accommodate those partial operations into the total framework of equational logic: for instance, we treat composition as a family of operations $\circ_{s,s'}$ with $s, s_\prime \in S$. Their meaning is the following. Composition expects two arguments of sort s and s' respectively, and they can be a combination of functions, inverse-finite functions, and homomorphisms. For composition to be well-defined, the left-hand side of s must match the right-hand side of s'. Thus, supposing $s, s' \in S_1$ we have, for instance, $s = (p \rightarrow q)$ and $s' = (r \rightarrow p)$. The result of composing functions of sort s and s' is a function of sort $(r \rightarrow q)$. Hence we let $\circ_{s,s'} \in \Sigma_{ss',(r \rightarrow q)}$. For other suitable combinations of s and s', we see to it that the composition of an inverse-finite function with a function produces a function, the composition of two homomorphisms is a homomorphism, etc. This means that we also have, for instance, $\circ_{s,s'} \in \Sigma_{ss',(p \rightarrow n)}$ where $s = (m \hookrightarrow n)$ and $s' = (p \rightharpoonup m)$, which expresses that the composition of a homomorphism with an inverse-finite function is a function. It should be clear that any meaningful composition, i.e., for any suitable s and s', can be accommodated in this way. The same holds for $\langle \_, \_ \rangle_{s,\ s'}$ which can be defined similarly. As for $\alpha_{s,s'}$, the only meaningful combinations are of the form $s = (p \rightarrow m)$ and $s' = (p \rightharpoonup q)$, and then we have of course $\alpha_{s,s'} \in \Sigma_{ss',(q \rightarrow m)}$.

Now we let the set E of equations over Σ be families of equations defined according to the following recipe. Take for example identity (6). According to the treatment of composition and aggregation above, it corresponds to the following family of equations:

$$\alpha_{(p \rightarrow m),(p \rightarrow r)}(v, z \circ_{(q \rightarrow r),(p \rightarrow q)} w) = \alpha_{(q \rightarrow m),(q \rightarrow r)}(\alpha_{(p \rightarrow m),(p \rightarrow q)}(v, w), z),$$

where p, q, r range over P and m ranges over M. It should be clear that all relevant equations of Sections 2 and 3 (i.e., Equations (1), (2), (7), and (8)) can be treated similarly.

From Meinke and Tucker (1992) we now know that the initial algebra for the equational class $Alg\ (\Sigma, E)$ of models of the equational theory E exists: it is the algebra consisting of congruence classes $[t\,]$ of terms t over Σ, where two terms are congruent if the one can be derived from the other using equations in E together with the deduction rules of equational logic (i.e., reflexivity, symmetry, transitivity, and substitution rules).

It should be clear that any Σ-algebra D that respects the sets-and-functions interpretation of Sections 2 and 3 is a member of the equational class $Alg(\Sigma, E)$. The sets-and-functions interpretation is formalized by choosing appropriate carriers D_s $(s \in S)$ for D where, for instance, $D_{(j \to o)}$ is the set of inverse-finite functions from a set j_D of jobs to a set o_D of organizations. Also, appropriate monoids and monoid operations must be chosen for each $m \in M$, for instance letting $\mu_{\alpha,D}$ (i.e., the interpretation of μ_α in D) be addition over $\mathbb{R}$, as each other constant symbol must be given appropriate meaning, choosing a particular combination of entity-value pairs for the variable $salary_{\mathrm{D}} \in D(j \to a)$ for instance. Finally, the interpretation of the operation symbols in D is then taken from Sections 2 and 3, for instance letting $\circ_{(p \to h),(j \to p),D}$ be the composition of two inverse-finite functions taken from $D_{(p \to h)}$ and $D_{(j \to p)}$ respectively.

7. References

Barr, M. and Wells, C. (1999). Category Theory for Computing Science. Montréal: Les Publications CRM.

CMF (2011). The Common Metadata Framework: Part B – Metadata Concepts, Standards, Models and Registries. Technical Report, UNECE.

Codd, E.F. (1969). Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks. Technical Report RJ599, IBM Thomas J. Watson Research Center.

Codd, E.F. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13, 377–387.

Elmasri, R. and Navathe, S.B. (1989). Fundamentals of Database Systems. Redwood City, CA: The Benjamin/Cummings Publishing Company.

Engelfriet, J. and Gelsema, T. (1999). Multisets and Structural Congruence of the pi-Calculus with Replication. Theoretical Computer Science, 211, 311–337.

Everitt, B.S. (2002). The Cambridge Dictionary of Statistics. Cambridge: Cambridge University Press.

Fraenkel, A.A., Bar-Hillel, Y., and Levy, A. (2001). Foundations of Set Theory. Amsterdam: Elsevier Science.

Gelsema, T. (2008). General Requirements for the Soundness of Metadata Models. Joint UNECE/Eurostat/OECD Work Session on Statistical Metadata (METIS), Luxembourg, 9–11 April.

Gelsema, T. (2010). Universal Properties of Aggregation. Technical Report DPK-2010-02-08-TGSA, Statistics Netherlands.

Goguen, J.A., Thatcher, J.W., Wagner, E.G., and Wright, J.B. (1977). Initial Algebra Semantics and Continuous Algebras. Journal of the ACM, 24III, 68–95.

Goguen, J.A., Thatcher, J.W., and Wagner, E.G. (1978). An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types. Current

Trends, in Programming Methodology, Volume IV: Data Structuring, R.T. Yeh (ed). Englewood Cliffs, NJ: Prentice-Hall.

Goguen, J.A. and Malcolm, G. (1996). Algebraic Semantics of Imperative Programs. Cambridge, MA: The MIT Press.

Grillet, P.A. (2001). Commutative Semigroups. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Gunter, C.A. (1992). The Semantics of Programming Languages: Structures and Techniques. Cambridge, MA: The MIT Press.

Klop, J.W. (1992). Term Rewriting Systems. Handbook of Logic in Computer Science, Vol. II: Background; Computational Structures, S. Abramsky, M. Gabbay, and T. Maibaum (eds). Oxford: Oxford Science Publications.

Mac Lane, S. (1998). Categories for the Working Mathematician. New York: Springer.

Meinke, K. and Tucker, J.V. (1992). Universal Algebra. Handbook of Logic in Computer Science, Vol. I: Background; Mathematical Structures, S. Abramsky, M. Gabbay, and T. Maibaum (eds). Oxford: Oxford Science Publications.

Pierce, B.C. (2002). Types and Programming Languages. Cambridge, MA: The MIT Press.

Pursiainen, H. (2008). Consistency in Aggregation, Quasilinear Means and Index Numbers. Technical Report 244, Helsinki Center of Economic Research.

SDMX (2011). SDMX Standards, Section 2, Information Model: UML Conceptual Design. Technical Report. Available at http://sdmx.org/.

Sundgren, B. (1973). An Infological Approach to Data Bases. PhD thesis, University of Stockholm.

UNECE Secretariat (2011). Final Report of the Workshop on Statistical Metadata (METIS), UNECE.

Vale, S. (2009). Generic Statistical Business Process Model. The METIS Workshop on the Statistical Business Process and Case Studies, Lisbon, 11–13 March.

van der Veen, G. (2011). Strategic Vision of the High-Level Group for Strategic Development in Business Architecture in Statistics. Paper presented at the 59th plenary session of the conference of European Statisticians (CES). Geneva, 14–16 June.

Wirsing, M. (1994). Algebraic Specification. Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, J. van Leeuwen (ed). Cambridge, MA: The MIT Press.

Received January 2011
Revised November 2011