Concept Description Language for Statistical Data Modeling

Tiziana Catarci (1), *Giovanna D'Angiolini* (2), *Maurizio Lenzerini* (1)

(1) Dipartimento di Informatica e Sistemistica,
Università degli Studi di Roma "La Sapienza"
Via Buonarroti 12, 00185 Roma, Italia
(2) Istituto Nazionale di Statistica (ISTAT)
via C. Balbo 16, 00185 Roma, Italia

Abstract

In this paper we describe a new language for statistical data modelling, which offers a general framework for the representation of elementary and summary data. There are three main characteristics of the language: 1) the types of modeling primitives of the language are particularly suited for representing objects from a statistical point of view; 2) the language includes a rich set of structuring mechanisms for both elementary and summary data, which are given a formal semantics by means of logic; 3) the language is equipped with specialized inference procedures, allowing for performing different kinds of checks on the representation.

1 Introduction

In the recent years, several approaches have been proposed for modelling large amounts of data from a statistical point of view [BD88,SW85,Su83,Gh86,OO85, OOM87].

A common characteristic of such approaches is to model elementary data by usual data models, then to define particular representation structures for summary data (i.e. data obtained by grouping elementary data, and by applying statistical operators -- such as total, average, percentage -- to such groupings). It follows that an integrated language for defining both elementary and summary data in a suitable way for statistical applications does not exist. Moreover, the proposed data models are often lacking in formalization, and do not provide the designer with powerful deductive capabilites.

Proceedings of the 16th VLDB Conference
Brisbane, Australia 1990

It is our opinion that statisticians would benefit from a design language offering both rich structuring mechanisms and inference procedures. In fact, in order to perform statistical analysis, the statistician gets the data of interest either by an ad hoc survey, or by manipulating pre-collected data, often extracted from very large data bases. In both cases, the design activity may require organizing a large collection of data, and checking if a certain amount of information is derived from the collected statistical data.

In this paper we present a new language for statistical data modelling, aiming to overcome the above mentioned drawbacks of existing proposals, and to offer a general framework for the representation of èlementary and summary data. There are three main characteristics of our proposal: 1) the types of modeling primitives of the language are particularly suited for representing objects from a statistical point of view; 2) the language includes a rich set of structuring mechanisms for both elementary and summary data, which are given a formal semantics by means of logic; 3) the language is equipped with specialized inference procedures, allowing for performing different kinds of checks on the representation.

We believe that the above characteristics makes the language an extremely useful tool during the phase of data modeling. In particular, by implementing the deductive machinery associated with the language, the designer may not only model the real world of interest by means of a rich set of linguistic primitives, but also ask the system to perform several checks which can be effectively used for controlling the design process.

In the tradition of the object-centered languages for knowledge representation (see [BS85, NS89]), our language allows both concepts and relationships to be described. A concept is an abstraction of a set of real world entities, whereas a relationship is an association between two concepts. Both concepts and relationships can be described by means of a rich set of constructors,

Work partially supported by Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of CNR, Tema INFOKIT.

starting from a set of primitive concepts (i.e. concepts which are described simply by a name). Due to this property, languages of this kind have been called *concept description languages*. Concept description languages originated with Brachman's KL-ONE [BS85], and grew out of research in semantic networks and frame-based systems. One of the main concerns of the work on such languages, has been to define suitable inference procedures for reasoning about concept descriptions. Recent works present a detailed analysis of the computational complexity of such reasoning procedures, depending upon the constructs used in the language [DH90]. We will mention in Section 4 that we can take advantage of such an analysis for characterizing the complexity of our language.

Using the constructors of the language, the statistician can define a basic collection of concepts, together with derived concepts, which are described by posing some restriction condition on the data in the schema. For instance, in a schema describing persons and their properties (age, sex, name, etc.), a new class may be defined based on a suitable restriction on the property "age", namely the class of persons with age < 14. Such a feature is particularly suitable for denoting subsets to be used as input for statistical analysis (see for example SAS procedures [Br81]).

The language is also equipped with suitable constructs for the definition of summary data, which are not present in the usual concept description languages. For instance, in the above schema, we may define the summary data "Average on age of persons partitioned by sex".

The paper is organized as follows. The basic notions related to statistical data modelling are presented in Section 2. In Section 3, we define the syntax and the semantics of the language. In Section 4, we deal with the basic deduction mechanisms. Finally, in Section 5 we show an example of statistical survey modeled using our language.

2 Basic Notions of Statistical Data Modeling

We are interested in statistical activities studying phenomena concerning finite populations (e.g. households, persons, enterprises, etc.).

The mean for observing a phenomenon on a finite population is the *statistical survey*. In a statistical survey, the *populations* of interest are defined, and the *phenomenon* is characterized by the values of a set of variables observed on each element of the populations. For instance, if the population is *Person*, the variables *Labour Force Status*, *Professional Status*, *Occupation* , *Wage* etc., may be chosen to characterize the phenomenon *Employment*.

We use the word *statistical unit* to mean an abstraction of a set of real world objects, called its instances, which form an enumerable class. The basic property of a statistical unit is the possibility of enumerating its instances and calculating totals on it, obtaining summary data.

In a statistical survey, populations are always described as statistical units. For example, *Person* is a statistical unit, on which one could be interested in calculating the summary data *Total of Wages of Persons by Professional Status* and *Number of Persons by Group of Wage*.

Besides populations, statistical units are also used to represent some kind of phenomenon, provided that the phenomenon can be described in terms of an enumerable class. For instance, the phenomenon *Vacation of Persons* can be described as a statistical unit, since we may calculate, for example, the *Total Expenditure of Persons for Vacations* and the *Number of Vacations by Vacation-Site*. On the contrary, the phenomenon *Employment* cannot be considered as a statistical unit.

The properties of an enumerable class represented by a statistical unit are modeled by means of *variables*. Example of variables describing properties of *Person* are: *Age*, *Sex*, *Personal-income*, *Address*, *Occupation*, etc. We distinguish between qualitative and quantitative variables. The domain of a qualitative variable is a set of states, which can be associated with the elements of the enumerable class. The domain of a quantitative variable is a set of numerical values, representing the observed measures of a measurable characteristic. It is worth noting that also the domain of a qualitative variable may be a set of numbers (e.g. *ATECO-Code* for *Economic-Activities*), but such numbers are not measures. It follows that only quantitative variables may be used as *summary variables*. A summary variable is such that it makes sense to summarize it with respect to a statistical unit. Examples of summary variables are *Personal-Income*, *Expenditure for Vacations*, etc, whereas *Sex*, *Occupation*, *Professional-Status*, etc. are examples of non-summary variables.

Since variables describe properties, they are defined as functions linking a statistical unit to a given doamin. For example *Sex* links *Person* to *{Male, Female}*. Statistical units are linked each other by *functions* or *relations* (notice that not every function is a variable). For instance, the statistical units *Person* and *Vacation* are linked to each other by the relation *Enjoy*. Links between codomains are realized by 1:n functions. For example, the codomain of *District* is linked to the one of *City*, where each city may be intended as a grouping of several districts.

In a statistical survey, it may be interesting to single out subsets of statistical units, called *statistical sub-units*. A statistical sub-unit S is described starting from a statistical unit U, and specifying the condition that the instances of U must satisfy in order to be instances of S.

The condition may be defined either on variables of U, or on statistical units which are linked to U by functions or relations. For example, *Person with age<14* is obtained from the statistical unit *Person* using its variable *Age*; *Household with unemployed head of household* is obtained from *Household* using the variable *Labour-Force-Status* and the statistical unit *Person*. Statistical sub-units may also be obtained by set operations on other sub-units.

The ultimate goal of a statistical survey is to compute summary data concerning the phenomenon of interest. The summary data of a statistical survey are specified by means of statistical goals. A *statistical goal* is the specification of how to compute a value using statistical operators. Two main kinds of goals may be singled out, depending upon the type of the statistical operator.

The first kind of goals refers to either totals computed with respect to summary variables, or count of elements of a statistical unit. These are defined by using statistical units partitioned according to the values of some variables. For example, referring to the statistical unit *Person* and the variables *Sex*, *Age*, and *Personal-income*, we may calculate the goal *Personal-income by Sex and Age*, where *Personal-income* is the summary variable and *Person* is partitioned using the values of the cartesian product of the codomains of *Sex* and *Age*. Moreover, if *Vacation* is a statistical unit with variable *Expenditure*, we may calculate the goal *Total Expenditure for Vacations of Persons by Sex and Age*, where *Expenditure* is the summary variable and *Vacation* is partitioned using the variables *Sex* and *Age* linked to it indirectly by the statistical unit *Person*. The result of such computations is an ordered set of numerical values, each one corresponding to an instance of the cartesian product of the codomains of the specified variables.

It is worth noting that goals of this kind may be derived from each other, under particular conditions on their partitioning variables. For example, *Number of Persons by Sex* is derivable from *Number of Persons by Sex and Age.*

The second kind of goals refers to computations obtained by using statistical operators such as average, percentage, square mean error, codeviance, etc. Due to the semantics of such operators, the goals of this kind can only defined on previously defined goals. For example, starting from the goals *Personal-income by Sex and Age* and *Number of Persons by Sex and Age*, we may define *Average on Personal-income by Sex and Age.*

3 The Language

In this section we describe the syntax and the semantics of a concept description language for the specification of statistical surveys. The language provides suitable constructors for describing statistical units, variables, statistical sub-units, aggregates and goals. The presentation is organized as follows. In the first subsection, we describe the basic mechanisms for building concept and relationship descriptions. In the second subsection we deal with the description of both aggregates and statistical goals. Finally, in the third subsection we show how to specify all the knowledge about a statistical survey by means of the language.

3.1 Concept formation

We assume the existence of two countably infinite sets *P*, *R* of concept and relationship symbols, respectively. The set *R* is partitioned into two subsets, one containing binary relation symbols, called roles, and one containing function symbols. The symbols in *P* and *R* describe primitive concepts and relationships, i.e. elementary concepts and relationships which are simply described by means of a name. We assume that *P* contains two distinguished concept symbols, namely Empty and T, denoting the empty and the universal concept respectively, together with suitable concept symbols denoting the usual domains of values used in programming languages, such as INTEGER, REAL, BOOLEAN, etc..

Concept and relationship descriptions are built out of concept and relationship symbols in *P* and *R* according to the following syntax (P denotes a primitive concept symbol, Q a primitive relationship symbol, C,D,C_1,...,C_n concept descriptions, R,S,T relationship descriptions, F_1,...,F_m function symbols, v_1,...,v_n values and n an integer):

```
C,D    --->  P | not P | C and D | C or D |
       some R.C | all R.C | atleast n.R |
       atmost n.R | F.C | setof C |
       [F1: C1,...,Fm: Cm] | (v1,...,vn)
S,T    --->  Q | S∘T | S^-1 | S:C
```

The set of all the concept descriptions built up from *P* and *R* will be denoted by *L(P,R)*. Notice that, in the above syntax, v_1,...,v_n denote values. In fact, we assume that a symbol is available for each value (of type integer, boolean, real, etc.) of interest. As a notational convenience, we also assume that sets of values which are intervals of an ordered domain can be written in the form: $v_1..v_n$.

The intuitive meaning of the operators *not*, *and* and *or* is simply set complement, set intersection and set union, respectively. Notice that *not* can only be applied to primitive concepts. The operators *setof* is used to describe the set of all the collections of instances of a given concept. The concept [F_1: C_1,...,F_m: C_m] denotes the set of all the (labeled) tuples where each component is labeled by the function symbol F_i and has an instance of C_i as value. The operators *some* and *all* are used to describe concepts on the basis of their linking to roles: intuitively, *some* R.C denotes the set of objects which are linked by the role R to at least one instance of the concept C, whereas *all* R.C denotes the set of objects which are linked by R to objects that are instances of C (analogously for F.C, when dealing with functions). The operators *atleast* and *atmost* are used to constrain the number of links of type R.

With regard to relationships, the symbol "$\circ$" denotes composition, the symbol $^{-1}$ is used to denote the inverse of a relationship, and the symbol ":" denotes the restriction of a relationship to those pairs whose second component is an instance of the concept C.

More formally, the semantics of the language state that each concept description in $\mathcal{L}(P,R)$ is interpreted as a subset of a universe U, according to suitable rules defining the meaning of the operators. The universe U is a set of structured objects, built up from a basic domain D of elementary objects, containing all the values of interest, and defined as the smallest set containing D and such that, if $u_1,...,u_n$ are in U, and $F_1,...,F_n$ are function symbols of $\mathcal{L}(P,R)$, then both $<F_1:u_1,...,F_n:u_n>$ (labeled tuple) and $\{u_1,...,u_n\}$ are in U.

Let $I(\cdot)$ be a function mapping each concept to a subset of U, and each relationship to a subset of $U \times U$. I is called an interpretation for $\mathcal{L}(P,R)$ if the conditions specified in Table 1 hold.

As an example of concept description, consider the concept *E* defined as *Employee with at least 2 houses and all of whose children are students*. Such a concept can be described in our language as

Employee *and* (*atleast* 2.PossessHouse) *and* (*all* Child.Student)

where Employee and Student are concepts, and PossessHouse, Child are roles.

Now let us consider the interpretation I with basic domain D={a,b,c,d,e,f,g,h,m,p} assigning {a,b} to Employee, {c,d,e,f} to Student, {<a,c>, <a,d>} to Child, and {<a,g>,<a,h>, <a,m>, <b,p>} to Possess-of-House. It is easy to verify that I assigns {a} to the above concept *E*.

3.2 Aggregates and statistical goals

An aggregate is a specification of how to classify all the instances of a given concept, called the target concept of the aggregate, according to their values with respect to a collection of properties. For example, an aggregate can be defined classifying Persons according to their sex and age.

Aggregates are classified into simple and complex. A simple aggregate has the form

$$[| s_1,...,s_n |]$$

I(Empty)	$= \emptyset$
I(T)	$= D$
I(*not* P)	$= D - I(P)$
I(C *and* D)	$= I(C) \cap I(C)$
I(C *or* D)	$= I(C) \cup I(D)$
I(*some* R.C)	$= \{ x \in U \mid$ there exists $<x,y> \in I(R)$ such that $y \in I(C) \}$
I(*all* R.C)	$= \{ x \in U \mid$ for each $<x,y> \in I(R)$ it holds that $y \in I(C) \}$
I(*atleast* n.R)	$= \{ x \in U \mid$ there exist at least n y such that $<x,y> \in I(R) \}$
I(*atmost* n.R)	$= \{ x \in U \mid$ there exist at most n y such that $<x,y> \in I(R) \}$
I(F.C)	$= \{ x \in U \mid <x,y> \in I(F)$ and $y \in I(C) \}$
I([F_1: C_1,...,F_m: C_m])	$= \{ <F_1: v_1,...,F_m: v_m> \in U \mid v_i \in I(C_i), i=1,...,m \}$
I(*setof* C)	$= \{ \{v_1,...,v_k\} \in U \mid v_i \in I(C), i=1..m \}$
I((v_1,...,v_n))	$= \{v_1,...,v_n\}$
I(S $\circ$ T)	$= \{ <x,y> \in U \times U \mid$ there exists $z \in U$ such that $<x,z> \in I(S)$ and $<z,y> \in I(T) \}$
I(S^{-1})	$= \{ <x,y> \in U \times U \mid <y,x> \in I(S) \}$
I(S:C)	$= \{ <x,y> \in U \times U \mid <x,y> \in I(S)$ and $y \in I(C) \}$

Table 1: Definition of interpretation

where $s_1,...,s_n$ are sets of values, often described as intervals, of the same domain N. In this case N is called the target of the aggregate. A complex aggregate has the form:

$$agg\ C * S_1.B_1 *...* S_n.B_n$$

where C is a concept, $S_1,...,S_n$ are relationships, and $B_1,...,B_n$ are simple aggregates whose target are codomains of variables. In this case, C is the target of the aggregate.

From a semantical viewpoint, a simple aggregate is a set of sets, whereas a complex aggregate of the form

$$agg\ C * S_1.B_1 *...* S_n.B_n$$

is defined as a set of labeled tuples. Each tuple corresponds to a combination of values for the specified properties, plus the set of all the instances of the target concept having such values of properties. More formally, given an interpretation I, the semantics of an aggregate is as follows (Target is a distinguished function symbol):

$$I([|s_1,...,s_n|]) = \{ I(s_1), ...,I(s_n) \}$$
$$I(agg\ C * S_1.B_1 *...* S_n.B_n) =$$
$$\{<S_1:v_1,...,S_n:v_n,\text{Target}:s> \mid v_1 \in I(B_1),...,v_n \in I(B_n),$$
$$s \text{ is the set of all the elements E of } I(C) \text{ such that } I(S_1).E \in v_1,...,I(S_n).E \in v_n \}$$

We assume that when in $S_i.B_i$ the specification of B_i is omitted, then the simple aggregate denoted by B_i is $[| (v_1),...,(v_n) |]$, where $v_1,...,v_n$ are all the values of the codomain of S_i. For example, the aggregate classifying persons with respect to Sex (whose codomain is {male,female}) and Age (partitioned into intervals 1..50,51..80), can be defined as follows:

(*agg* Person * Sex * Age.[|1..50,51..80|])

Suppose that I is the interpretation defined as follows:

I(Person) = {a,b,c,d}
I(Sex) = {<a,male>, <b,male>, <c,female>, <d,female> }
I(Age) = {<a,20>, <b,30>, <c,20>, <d,61> }

Then, I((*agg* Person * Sex * Age.[|1..50,51..100|])) =
{<Sex:{male},Age:{1..50},Target:{a,b}>,
<Sex:{male},Age:{51..100},Target:{}>,
<Sex:{female},Age:{1..50},Target:{e}>,
<Sex:{female},Age:{51..100},Target:{d}>}

A goal is a specification of a meaningful computation to be performed in the context of a statistical survey. In the definition of goals we make use of a set of operators which are usually considered in statistical data bases, such as COUNT, AVERAGE, etc.. Each operator has a set as argument and returns values of a distinguished type.

Goals are classified into simple and composite. A simple goal has the form:

(*goal* OP *on* R *of* AGG)

where OP in an operator, R is a summary variable, and AGG is an aggregate. Intuitively, the above goal is a specification of a collection of values, each one relative to the target component $M=\{c_1,...,c_p\}$ of an instance of AGG; each value is obtained by applying the operator OP to the set of values corresponding to the property R of $c_1,...,c_p$. As a notational convenience, the specification of R can be omitted, in which case, the operator is intended to be applied directly to the set $c_1,...,c_p$.

More formally, given an interpretation I, the semantics of a simple goal is as follows (Result is a distinguished function symbol):

I((*goal* OP *on* R *of* AGG)) = { [|Target: c, Result: x|] | there is $<S_1:v_1,...,S_n:v_n$, Target:c> in I(AGG), and x is the result of applying OP to the set of values corresponding to I(R.c) }

I((*goal* OP *on* AGG)) = { [|Target: c, Result: x|] | there is $<S_1:v_1,...,S_n:v_n$, Target:c>in I(AGG), and x is the result of applying OP to the set of values corresponding to I(c) }

For example, referring to the interpretation I in the previous example, the semantics of the goal (*goal* COUNT *on* (*agg* Person * Sex * Age.[|1..50,51..100|])) is:

{ <Target:{a,b}, Result:2>, <Target:{}, Result: 0>, <Target:{c}, Result: 1>, <Target:{d}, Result: 1> }

A composite goal is simply an application of an operator to a collection of arguments which are goals themselves. The form of a composite goal is

(*goal* OP *on* G_1 ... G_n).

and its semantics is easily obtained from the semantics of both the operator OP and the arguments $G_1,...,G_n$.

3.3 Specification

A specification is a collection of statements describing all the relevant knowledge about a statistical survey. It is constituted by three parts: in the first part, names are assigned to the various objects involved in the survey. Moreover, concepts are classified into statistical units and statistical sub-units, whereas relationships are classified into variables, roles and functions. In the second part, constraints are declared describing the definition of statistical sub-units, the conditions that units must satisfy, and the domain and codomain of variables, roles and functions. Finally, the third part is devoted to the definition of both aggregates and statistical goals. The syntax of a specification is shown in Table 2.

	StatisticalSurvey N;
Units	unit-name,...,unit-name
Sub-units	sub-unit-name,...,sub-unit-name
Variables	var-name,...,var-name
Roles	role-name,...,role-name
Functions	fun-name,...,fun-name
	Constraints
	γ_1 ... γ_n
	Aggregates
	α_1 ... α_n
	Goals
	τ_1 ... τ_n

Table 2: Syntax of a specification

The constraint part is constituted by a set of constraint assertions, each one specifying a semantic condition on concepts or roles. There are three kinds of constraint assertions:

1) A constraint can be used for fixing the domain and the codomain of a relationship. In this case, the statement has the form:

$$R: D_1 \times D_2$$

where D_1 and D_2 are concept descriptions. Notice that this mechanim allows for the specification of the domain and the codomain of each variable.

2) A constraint can be used to state necessary conditions that a unit must satisfy. In this case the

statement has the form:

$$P \textit{ isa } \alpha$$

where P is the name of a unit, and α a concept description. For example, the assertion (Person *isa* Age.(1..100)), states that every instance of Person must possess an age, whose value is in the range 1..100. Due to the expressive power of the concept description language, the above is a quite powerful tool for specifying constraints.

3) A constraint can be used to define sub-units. In this case the statement has the form:

$$P \textit{ def } \alpha$$

where P is the name of a sub-unit, and α a concept description. For example, the assertion (Young_father *def* (Person *and* (*some* Child.T) *and* Age.(1..24))) defines the sub-unit Young_father in terms of the role Child and the function Age.

The semantics of the constraint part is simply given by specifying the conditions under which an interpretation I satisfies a constraint. I satisfies a constraint of the form

$$R: D_1 \times D_2$$

if I(R) is a subset of $I(D_1 \times D_2)$. I satisfies a constraint of the form

$$P \textit{ isa } \alpha$$

if I(P) is a subset of $I(\alpha)$. Finally, I satisfies a constraint of the form

$$P \textit{ def } \alpha$$

if $I(P)=I(\alpha)$. I is said to be a model of the specification σ if I satisfies all the constraints of σ.

In the aggregate part (resp. goal part) every α_i (resp. τ_i) refers to an aggregate (resp. goal) definition, which has the form:

$$\alpha \textit{ def } \delta$$

where α is the name of the aggregate or the goal being defined, and δ is the corresponding description, formed according to the syntactic rules given in Subsection 3.1.

4 Reasoning about a Specification

The language described in the previous section allows for expressing the knowledge about a statistical survey. Based on the formal definition of the semantics of the language, we can devise specialized techniques for reasoning about a specification. We will show in this section that such techniques constitute a valuable support to the designer of statistical data schemas.

Notice, first of all, that a specification may suffer from a number of anomalies, due to a wrong usage of the constructs of the language. For example, if both the constraint assertions

Male *isa not* Female

Joung_Person *def* Female *and* Male *and* Age.1..20

are in a specification σ, then in all the models of σ, the set of instances of Joung_Person is empty. Obviously, in this trivial example, the problem can be easily singled out, but it is quite clear that in more complex situation, it may not be evident that a concept is invariably empty in all the models of the specification.

In order to formalize the above considerations, we now introduce the notions of consistency and inconsistency of concepts and specifications. A concept C is said to be inconsistent (consistent otherwise) in a specification σ, if for each model M of σ, it holds that $M(C)=\emptyset$. A specification σ is said to be consistent (inconsistent otherwise) if no concept of σ is inconsistent. Also, we say that the concept C is subsumed by the concept D in a specification σ, if for each model M of σ, M(C) is a subset of M(D).

Consistency can be thought of as the basic property to be checked about a specification. More generally, the designer may want to verify whether a certain property holds in all the models of a specification σ. This can be done by providing the designer with a deductive method that, given a specification σ and an assertion α of one of the forms described for constraints, checks whether

$$\sigma \models \alpha,$$

i.e. whether α is satisfied by every model of σ. Notice that, for example, the consistency of the concept C may be checked by testing whether ($\sigma \models$ C *isa* Empty), and subsumption can be checked by testing whether ($\sigma \models$ C *isa* D).

For the sake of brevity, we do not delve into the details of such a method for our language. The interested reader is referred to [CDL90], where we describe a sound and complete algorithm for performing the above mentioned deductions on a specification. The algorithm is based on a general technique developed in the context of concept description languages (see [SS90]). By using complexity results concerning such languages (see [SS90, DLM90]), in [CDL90] we characterize the computational complexity of the deduction method for our language, which is exponential in general, and single out syntactic restrictions of the language in order to get tractability.

Other reasoning facilities are provided in our language, concerning aggregates and goals.

An aggregate A is said to be derivable from an aggregate B in a specification σ, if there exists a total one-to-many correspondence R from the instances of A and B such that, for each model M of σ, for each x in M(A), $x=y_1 \oplus \ldots \oplus y_n$, where $R(x)=\{y_1,\ldots,y_n\}$, and $\oplus$ denotes an operator between tuples, defined as follows: $<F_1:v_1,\ldots,F_n:v_n, \text{Target}:c> \oplus$

$<F_1:w_1,\ldots,F_n:w_n, \text{Target}:d> =$

$F_1:v_1 \cup w_1,\ldots,F_n:v_n \cup w_n, \text{Target}:c \cup d>$.

In [CDL90], we describe sufficient conditions for derivability of aggregates. These may be used for verifying what we call consistency of aggregates: an aggregate A is said to be inconsistent in a specification σ, if for each model M of σ, for each tuple x in M(A), the Target component of x is empty. Notice that A=(*agg* C *

$S_1.B_1 *...* S_n.B_n$) is inconsistent in σ, if and only if (*agg* Empty $*$ $S_1.B_1 *...* S_n.B_n$) is derivable from A. Therefore, we can use derivability for checking the consistency of aggregates.

An analogous notion of derivability can be defined for goals, as described in [CDL90].

5. An Example of Statistical Survey

We describe an example of statistical survey concerning the employment condition of a sample of persons in Italy. We want to study the phenomenon *Employment* by means of the observation of several related factors. In particular, we consider the *Household* to which a person belongs, the *Habitation* in which one lives, together with information related to each person, such as *Sex*, *Age*, *Marital-Status*, *NoDegree*, etc.

The aim of such a survey is to obtain a set of summary data, namely: *Number of Employed Persons with Age <30 by Sex, Activity-Field, and Region*; *Number of Unemployed Persons by Age-Group, NoDegree, and Income-Group of the Household*; *Average on Income of the Households with at least one unemployed member by Region*, where a region is a geographical area having local administration, and partitioned into so-called provinces.

The statistical survey *Employment* is described in our language as follows.

<u>Statistical Survey</u> Employment;
<u>Units</u>
  Person, Household, Habitation;
<u>Sub-Units</u>
  PersonWithAge>14,EmployedPerson,
  UnemployedPerson,
  Unemployed-previously-employed;
<u>Variables</u>
  Sex, Age, Marital-status, Labour-Force-Status,
  NoDegree, Income, Rooms, Street, Number, City,
  Habitation-kind, Province, Region, Activity-field,
  Previous-employment;
<u>Roles</u>
  Possess;
<u>Functions</u>
  Member-of
<u>Constraints</u>
  Sex: Person $\times$ (M,F);
  Age: Person $\times$ (0..120);
  Marital-Status: Person $\times$ (Married, Widow, Divorced, Single);
  NoDegree: Person $\times$ (HighSchoolDegree, MasterDegree, PHDdegree);
  Income: Household $\times$ Integer;
  Rooms: Habitation $\times$ Integer;
  Street: Habitation $\times$ String;
  Number: Habitation $\times$ Integer;
  City: Habitation $\times$ NameofCity;
  Habitation-kind: Habitation $\times$ (primary, secondary);
  Province: NameofCity $\times$ NameofProvince;
  Region: NameofProvince $\times$ NameofRegion;
  Activity-field: (EmployedPerson *or* Unemployed-previously-employed) $\times$ (Industry, Agriculture, Services);
  Labour-Force-Status: PersonWithAge>14 $\times$ (employed, unemployed);
  Previous-employment: UnemployedPerson $\times$ Boolean;
  Possess: Household $\times$ Habitation;
  Members-of: Household $\times$ Person;
  Person *isa* Sex.T;
  Person *isa* Age.T;
  Person *isa* Marital-status.T;
  Habitation *isa* Rooms.T;
  Household *isa* Income.T;
  Habitation *isa* Street.T;
  Habitation *isa* Number.T;
  Habitation *isa* City.T;
  Habitation *isa* Habitation-kind.T;
  Household *isa* Member-of.T;
  Person *isa* Member-of^{-1}.T;
  PersonWithAge>14 *def* Person *and* Age.(14..99);
  EmployedPerson *def* PersonWithAge>14 *and* Labour-Force-Status.(employed);
  UnemployedPerson *def* PersonWithAge>14 *and* Labour-Force-Status.(unemployed);
  Unemployed-previously-employed *def* UnemployedPerson *and* Previously-employment.(true);
<u>Aggregates</u>
  Agg1 *def* (*agg* (EmployedPerson *and* Age.(14..29)) $*$ Sex $*$ Activity-field $*$ Member-of $\circ$ Possess: (Habitation *and* Habitation-kind.(primary)) $\circ$ City $\circ$ Province$\circ$ Region);
  Agg2 *def* (*agg* (UnemployedPerson $*$ Age.[|14..29,30..50,51..70,71..99|] $*$ NoDegree $*$ Member-of $\circ$ Income.[|1..10, 11..50,51..100,101..500,501..Max|]);
  Agg3 *def* (*agg* (Household *and* (*some* Member-of^{-1}.UnemployedPerson) $*$ Possess: (Habitation *and* Habitation-kind.(primary)) $\circ$ City $\circ$ Province $\circ$ Region);

Goals

Goal1 *def* (*goal* COUNT *on* Agg1);
Goal2 *def* (*goal* COUNT *on* Agg2);
Goal3 *def* (*goal* COUNT *on* Agg3);
Goal4 *def* (*goal* SUM *on* Income *of* Agg3);
Goal5 *def* (*goal* QUOTIENT *on* (Goal4, Goal3));

It is worth noting that, in the above schema, the goals Goal3 and Goal4 are not explicitly requested in the survey; they are defined as intermediate goals, in order to properly define Goal5. In this case, the derivability relation between goals is evident from the definition. As claimed in Section 2, other derivability relations between goals can be inferred by reasoning on their definitions.

Notice that meaningful aggregates are not necessarily given a name. For example, *Age-Group* is implicitly defined in Agg2 as [|14..29,30..50,51..70,71..99|] (analogously for *Income-Group*).

After the description of the statistical survey, an analysis phase may take place, in order to check some meaningful properties of the specification. For example, the designer may want to check, first of all, whether the specification is consistent. The concepts involved in the specification are, therefore, checked one by one for consistency. The result, in this case, is that the specification is consistent. It is easy to verify that the same holds for all the aggregates defined in the survey.

Moreover, the designer may want to check whether other goals can be derivable. For example, suppose that we become, later on, interested in the goal *Number of Unemployed-previously-employed by Age-Group, and Income-Group of the Household.* This can be done by specifying the goal:

(*goal* COUNT *on* (*agg* (Unemployed-previously-employed * Age.[|14..50,51..99|] * Member-of ∘ Income.[|1..10, 11..50,51..100,101..500,501..Max|])

Now, by using the facilities briefly described in Section 4, the designer may be acquainted with the fact that the aggregate specified in the above goal is derivable from the Agg2. Therefore, she/he can conclude that such a goal is derivable from Goal2.

6 Conclusions

We have described a new approach to statistical data modelling, based on a concept description language, which is formally defined in terms of logic, and is equipped with specialized inference procedures.

Such inference procedures provides the designer with an extremely useful tool during the phase of data modeling. The designer may not only model the real world of interest by means of a rich set of linguistic primitives, but also ask the system to perform several checks which can be effectively used for controlling the design process.

We have used the language in some applications concerning the Italian Census, and, based on the encouraging results, we are now implementing a complete system based on the language.

References

[Br81] Bragg A.W., "Data Manipulation Languages for Statistical Databases - The Statistical Analysis System (SAS)", Proc. of the First LBL Workshop on SDB Management: 147-150, 1981.

[BD88] Batini C., Di Battista G., "Design of Statistical Databases: A Methodology for the Conceptual Step", *Information System*, 13(4): 407-422,1988.

[BS85] Brachman R.J., Schmolze J.G., "An Overview of the KL-ONE Knowledge Representation System", *Cognitive Science*, 9(2): 171-216, 1985.

[CDL90] Catarci T., D'Angiolini G., Lenzerini M., "Concept Description Language for Statistical Data Modelling", Technical Report, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Italia, 1990.

[DH90] Donini F., Lenzerini M.,, Nardi D., Hollunder B., Nutt W., "The Complexity of Concept Description Languages", Technical Report, Universitat Kaiserslautern, West Germany, 1990.

[Gh86] Ghosh S.P., "Statistical Relational Tables for Statistical Database Management", *IEEE Transactions on Software Engineering*, SE-12(12): 1106-1116, 1986.

[NS89] Nebel B., Smolka G., "Representation and Reasoning with Attributive Descriptions", IWBS Report 81, IBM Deutschland, Stuttgart, West Germany, 1989.

[OO85] Ozsoyoglu G., Ozsoyoglu Z.M., "Statistical DatabaseQuery Languages", *IEEE Transactions on Software Engineering*, SE-11(10): 1071-1081, 1985.

[OOM87] Ozsoyoglu G., Ozsoyoglu Z.M., Matos V., ing Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions", *ACM Transactions on Database Systems*, 12(4): 566-592, 1987.

[SW85] Shoshani A., Wong H.K.T., "Statistical and Scientific Database Issues", *IEEE Transactions on Software Engineering*, 11(10), 1985.

[SS90] Schmidt-Schauβ M., Smolka G., "Attributive Concept Descriptions with Complements", To appear in Artificial Intelligence, 1990.

[Su83] Su S.Y.W., "SAM*: A Semantic Association Model for Corporate and Scientific Statistical Database", *Information Science*, 29: 151-199, 1983.