Computer Science Journal of Moldova, vol.**10**, no.1(28), 2002

Analytical space for data representation and interactive analysis

Alexandr A. Savinov

Abstract

In the article the notion of analytical space is introduced and its application to data representation and interactive analysis is studied. Analytical space is defined through membership relation among its elements where each element is characterised by its extensional and intensional. All data element properties are derived from this fundamental relation. Inference in analytical space is thought of as finding mapping from one subspace into another and is carried out through propagation of information by means of deaggregation and aggregation operations. The general goal of data analysis is defined to be simplifying the form of representation with simultaneous retaining most of the data semantics. An interactive data analysis procedure is proposed, which is based on selecting interesting subspaces in analytical space, finding mapping from this space into the range of values by means of inference and finally visualising this mapping in an appropriate form.

Key words: Data analysis, Data modelling, Hierarchical multidimensional data, Analytical space, Information aggregation, Data visualisation

1 Introduction

As large organisations accumulate huge databases and begin to realise the potential value of the information that is stored there the area of data analysis is being paid more attention. Although automatic

methods of intelligent data analysis being developed within data mining area [3] are rather powerful and can be applied to very large databases they cannot deal with ill structured and informal problems where both the search space and the goal cannot be precisely defined. For such tasks where data has highly complicated structure, which in addition may change in time, interactive methods of analysis are of especially great importance. To carry out such an interactive analysis appropriate data representation techniques should be used. For this purpose in this article we propose a data representation technique based on *analytical space* conception and describe how it can be used for interactive data analysis.

Any data can be characterised by two main notions: *multidimensionality* and *hierarchy*. Hierarchy appears where parent-child relationship among data elements has several levels, e.g., primitive attribute values can be grouped into classes, which themselves are used as values to characterise other elements. Multidimensionality is interpreted as a possibility to characterise one and the same element or a set of elements from several points of view, i.e., by means of several different elements, and appears due to existence of more than one parent. So far these two phenomena, hierarchy and multidimensionality, have been treated mostly independently, i.e., there exist methods for dealing with multidimensionality and there exist separate methods for dealing with hierarchy. Traditionally most methods of interactive data analysis are oriented on studying multidimensional structure of data. Only relatively recently new methods of data representation and analysis have appeared like OLAP [2], which try to manipulate hierarchies and multidimensionality in data simultaneously. In contrast to other data representation methods we derive both hierarchy and multidimensionality from one and the same underlying *membership relation* among elements of analytical space. The membership relation can be defined if each element is assigned a set of parents called the element *intensional* and a set of children called the element *extensional*. The fundamental membership relation allows us to explain such phenomena as characterisation by means of attributes and values, table representation, existence of levels of detail, common object attributes, common

attribute values etc. It is very important that all elements of analytical space are absolutely primitive and it is exclusively their relations with other elements what provides them some properties.

In addition to its flexible representation capability analytical space allows us to effectively define the notion of *logical inference*. By logical inference we mean finding a mapping from one subspace into another according to the semantics represented in analytical space. In other words, the problem of logical inference consists in calculating a function, which for each element from its domain finds an element from its range. The inference defined in such a way requires propagation of information through analytical space and can be carried out in two steps (for each element from the domain): first deaggregation from the source element down to the space lowest level, and second aggregation up to the target element from the function range.

The definition of analytical space as a set of elements along with some membership relation among them and inference as finding mapping from a domain to a range is rather formal and can be used in different application areas. In this article we demonstrate how the introduced notions can be applied to interactive data analysis. The goal of any data analysis is defined to be simplifying the form of representation simultaneously retaining most of the data semantics. For example, in data mining it might be rules that should be rather simple in form while keeping a great deal of information from data. In data analysis it is a mapping from one set of data elements into another that is meant by both simple and expressive representation of underlying semantics. However, because of highly complicated data structure and informal data interpretation the search for such an interesting function is carried out manually by a human analyst (although automatic methods might be also very useful). A general data analysis scenario introduced in the article consists of the following steps: data representation and definition of analytical space, selecting domain and range, calculating the function, selecting appropriate visualisation for the function.

The earlier ideas used in this paper were described in [6] where the conception of multidimensional hierarchical space was proposed. In this article this idea is further developed and applied to the task of

data representation and analysis.

2 Data Representation

2.1 Elements of Analytical Space

An *element* is a primitive building block of analytical space. It is an elementary indivisible unit, which does not have any internal structure. Each element taken by itself without any relationships with other elements of analytical space is by definition a primitive entity. It is a relation among elements in analytical space what brings a structure into each individual element. In other words, each element in isolation does not have any structure, which appears and is explained exclusively because of existing relations among all other elements.

In particular, at this fundamental level of abstraction we do not suppose that elements or objects of analytical space are characterised by any (internal) properties, which may take different values and thus distinguish elements from each other. Thus objects are distinguished exclusively by their position in analytical space, which is determined by the ties with other elements. In other words, the analytical space element semantics is always external in relation to each element. Yet at lower level of abstraction object properties can be derived from relationships among elements by means of an appropriate interpretation of different elements and their position. Such an interpretation in terms of object properties is more suitable for applications and will be described below in the article. In particular, different elements can be interpreted as tables, columns (properties), values, dimensions etc. The advantage of using the described abstraction where elements take their structure from relations is a possibility to generalise these and other conventional notions as well as manipulate them formally correctly and effectively.

2.2 Membership Relation among Elements

Analytical space is formally defined as a set of elements with membership relation among them. Thus for any pair of elements from analytical space we can ask if one of them is a member of another or not, i.e., if the relation is satisfied for these two elements. If one element includes another as its member then it can be formally considered a set in the sense of the set theory. Analytical space then can be considered a number of nested sets. Some sets may be empty (lowest level elements) while others are not members of any other element (highest level elements). Below we will show that all main properties of elements and many conventional notions can be derived from their position in relation to other elements in analytical space. It should be noted that analytical space itself can be viewed as the highest level element which directly or indirectly includes all other elements.

The membership relation among elements of analytical space is not arbitrary and has to satisfy some additional constraints. These constraints are highly natural and are derived from conventional requirements. The main requirement any analytical space structure must satisfy is the absence of transitive relations between elements, i.e., if $a \in b \in c$ then $a \notin c$. We also prohibit circular relations. In particular, an element cannot be a member of itself. Thus the structure of analytical space is equivalent to the acyclic non-transitive graph. Note that elements may well be members of simultaneously more than one other element and include as members more than one other element.

All elements, which are direct members of the given element are said to be its *extensional* or child elements. All elements the given element is included into as a direct member are said to be its *intensional* or parent elements. Below it will be shown that extensional is responsible for hierarchy while intensional is responsible for multidimensionality. Informally, the power of extensional expresses the element size while the power of intensional determines the element dimensionality.

To deeper understand the difference between these two notions it should be noticed that from logical point of view all elements of the extensional are combined with the OR-connective (logical sum). Dually

all elements of the intensional are combined with the AND-connective (logical product). Thus any element has two definitions: through its extensional and through its intensional. We assume that extensional is represented with the help of a data structure called *collection* which combines a set of elements by means of OR-connective while intensional is represented with the help of a so called *combination* which unites a number of elements by means of logical AND operation. Collection is normally implemented as an array. Conventional (in object oriented programming) object or record is an example of combination where fields are combined with logical AND. Collection can be thought of as a set of children while combination is a set of parents. To distinguish them we use $\{\}$ to write collection elements and $\langle\,\rangle$ for uniting combination elements.

Let us now consider how simple conventional elements of data representation can be modelled in analytical space (Fig. 1). Perhaps, the simplest and the most important notion (not only for data description) is that of *attribute* and its values. It is well know that this construction can be represented by an attribute element with extensional consisting of all its values elements. For example, two-valued attribute consists of two value elements while continuous attribute consists of an infinite number of values in its extensional. Obviously, the attribute-value construction can be easily generalised onto the hierarchical case where values may themselves play the role of attributes and be interpreted as groups or classes. If each element has only one parent then we obtain tree-like structure. The space is characterised by only one dimension but arbitrary number of levels of detail.

The main idea of using attributes and values is a possibility to characterise other objects by means of one set of values so that all objects are put into one space where they can be compared or managed in some other way. Dual to the notion of attribute is the notion of *object*. In contrast to attribute, object combines its values by means of logical AND connective, i.e., all its values constitute its intensional rather than extensional (Fig. 1). One practical difference between attributes and objects is that objects in most cases take a fixed number of values (although it is not formally necessary). This number essen-

tially determines the object dimensionality. For example, if an object is characterised by 3 fields then it is 3-dimensional element.

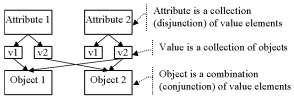

Figure 1. Duality of attribute and object

These two dual constructions can be easily and naturally combined into one where an object takes values of a number of attributes. It is the most important building block used to represent and analyse data. One very important formal property of this construction is that any object is always included into each of its attribute values, i.e., value is a container for objects it characterises. For example, the following relations are true: $\langle 11, 25\rangle \in 11$ and $\langle 11, 25\rangle \in 25$. In dual form it is obvious: $11 \in \langle 11, 25\rangle$ and $25 \in \langle 11, 25\rangle$. By increasing the number of attributes we can easily obtain conventional multidimensional data representation mechanism by means of tables (Fig. 2). However, the final form this method acquires if we add one special attribute the values of which are objects themselves. Indeed, the table representation mechanism (as well as other object approaches) supposes that rows are individual objects, which can be identified even with no columns at all, i.e., theoretically we may have a table with not columns where rows are still distinguishable. Such a 0-dimensional table can be represented as an attribute with the extensional consisting of all rows. In other words, the values of such a table-attribute are object references.

It is a subtle but very important moment that references are not object properties – they are object themselves. Object and its reference are equivalent notions just because there does not exist any other way of perceiving an object except for its reference. Moreover, object reference is the only thing we have, particularly, there is not such a notion

as real object as opposed to object reference. This fundamental role of references distinguishes them, e.g., from identifiers, which are considered user specific attributes just like any other attributes. Because of this property of references and one-to-one unique correspondence between references and rows we do not consider references as special values, which are taken by rows. Instead rows themselves are viewed as values.

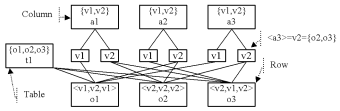

Figure 2. Multidimensional analytical space equivalent to table

The following aspects are important to notice. Attribute elements have the highest level in the membership hierarchy (except for analytical space itself) because they have empty intensional. The table element is formally equivalent to attributes so it is also at the top level (in the next section it will be shown that table represents lower level of detail in relation to attributes). The row (object) elements have the lowest level since they have empty extensional (although in general case it is not necessary, i.e., rows may have their own non-empty extensional). In this sense attributes and rows are opposite elements in the membership hierarchy of such a conventional multidimensional analytical space. Row objects may be members of maximum one attribute value, i.e., informally, attribute is intentionally defined so that objects may take only one or no of its values. (Although there may be more complex cases, e.g., multivalued attributes where it is possible.)

Objects are generally not required to take some concrete attribute value, i.e., to have a parent from all attribute extensionals. Formally any element may take as many parent elements, i.e., as many values, as needed for describing this element semantics (element semantics is

described exclusively externally through the ties with other elements). If an object does not belong to any value of some attribute then we say that it is not characterised by this attribute at all, i.e., the attribute is not relevant for the object. Yet if it is for some reason necessary to describe the object in terms of this attribute then the special *null* value is used, which is formally equivalent to empty set. The equivalence of null value to the absence of attribute for the object is rather important moment of the formalism. It is used, e.g., for equivalent analytical space transformations and logical inference. In other words, null value and its interpretation is not just a convention useful for applications. It is an important formal notion, which should not be mixed with such out of formalism notions like, e.g., unknown value, NA etc. For example, because of this the following identities are true: $\langle 11, null \rangle \equiv 11$ and $\langle null, 25 \rangle \equiv 25$ (essentially they can be viewed as a definition of the null element).

2.3 Inclusion and Orthogonality Relations

The general purpose of multidimensionality is representing and viewing objects from different points called dimensions. For example, one and the same set of objects looks differently if viewed from the sides of two attributes. Multidimensionality of data in analytical space is represented by *orthogonality relation* among elements (Fig. 3). Two elements (attributes) are said to be orthogonal if elements from their extensional (values) have common children (objects). Conventionally, attributes characterising one set of objects are mutually orthogonal.

It should be noted that orthogonality is defined in relation to some objects and it is normally used as a constraint when defining the structure of analytical space. For example, if we say that a set of objects from some table is characterised by two orthogonal attributes then it means that we must include each new object into one value for each attribute (yet, the final semantics is defined by the membership relation among elements).

In contrast to orthogonality, the main purpose of *inclusion rela-*

tion is representing hierarchy and viewing objects at different levels of detail or scale. One element is said to *include* (hierarchical relation) another element if its elements (values) consist of elements of another one (Fig. 3). Orthogonal elements represent completely incompatible entities while hierarchy is represented by elements, which are completely included into one another and represent levels of detail of one dimension. Just like for orthogonality the inclusion relation is derived from the underlying membership relation rather than defined independently. However, for practical purposes it is convenient to define elements of hierarchy in advance as a constraint imposed on the analytical space structure. For example, we might say that some attribute has a higher level of detail where numeric values are mapped into discrete intervals or classes.

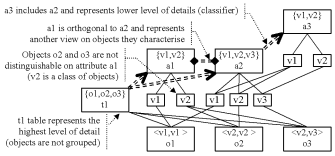

Figure 3. Orthogonality and inclusion relations

It is very easy to illustrate the inclusion relation already on simple tables. Indeed, the table element consists of individual objects (rows) and represents the highest level of detail (all objects are distinguishable by their reference). However, if we select one attribute then all objects can be broken into larger classes depending on what value each of them takes. Thus each class is represented by one attribute value and objects taking one value are not distinguishable. Obviously, there exist as many such lower levels of details as we have attributes for describing objects. The table element in this case is included into each attribute element.

In this sense attributes can be defined as classifiers for a set of objects.

The attributes themselves can be further classified by decreasing their level of detail. This procedure is frequently applied to attributes with a lot of values, which appear to have too high level of detail, e.g., numeric attributes. The values of attributes are grouped into classes where each class is represented as a value of classifier attribute.

Clustering is very similar to classification in that they both intended to decrease the level of detail. The main difference is that clustering as a grouping process is normally applied to complex multidimensional objects, which are characterised by many features while classification is used for grouping simpler objects like attribute values. Clusters breaking a set of objects into groups can be easily represented in analytical space by values of special attributes. Objects, which belong to one cluster, are members of the corresponding values.

The above examples were more or less simple and emphasised only one aspect of analytical space. Let us now consider more complex example where we illustrate how multidimensionality and hierarchy can be combined more tightly. We can notice that for any attribute its values can be thought of as normal objects. By standard definition any value has only one parent, which is common for all values of one attribute. In other words, the only characteristic of any value is that it belongs to some attribute element representing the corresponding feature. Yet we can suppose that the values themselves may be characterised by more than one feature. In this case their attribute plays a role of table (a container of objects) while attributes characterise the value elements by assigning different properties (Fig. 4). For example, the date attribute may be viewed as consisting of concrete indivisible objects representing concrete moments in time, which are used as values and characterise some set of objects. However, the dates themselves can be further (at lower level of detail) characterised by additional properties and grouped into the corresponding classes such as, e.g., day of week and month, which are mutually orthogonal. We can decrease the level of detail even more by grouping months into quarters or introducing additional characteristics of date like if it a holiday or not.

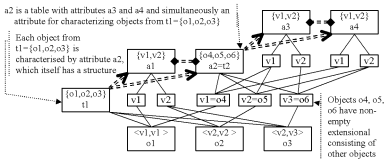

Figure 4. Structured multidimensional attributes

This example demonstrates that elements in analytical space cannot be always clearly referred to such wide spread notions as table, attribute, value, object, record, class etc. because many of them have intermediate position and complex membership relation with neighbours. What is important to notice is that many of them combine both multidimensional properties having more than one parent element and hierarchical properties having both parents and children.

Characterising objects by means of several values from different attributes is a conventional practice where all objects are characterised by the same set of attributes. The dual situation where objects from different extensionals are characterised by one and the same attribute(s) is much less obvious for understanding. For example, we may have a set of country objects characterised by some properties like name, area, religion and so on. Each country however can be viewed as consisting of demographic objects and economic objects both described by their own attributes set like age and profit, respectively. Thus all children of one country object are broken into two groups according to their involvement into two tables: demography and economy (Fig. 5). This is obviously dual to the situation where all parents of one object are broken into several groups depending on their involvement into one of the attributes. In other words, it may well happen that both parent objects and child objects do not constitute one set, i.e., they do not

have a common container element. Note that precisely the same situation appears if one attribute is used to characterise (classify) objects from two or more tables. In this case each its value may have children from more than one table. The country table can be viewed as such a common attribute which groups objects from demography and economy tables according to their country membership.

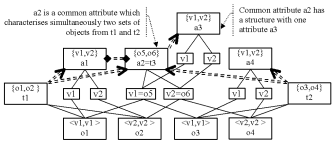

Figure 5. Common attribute with additional structure

2.4 Inference and Aggregation

Logical inference is rather wide area, which strongly overlaps with different approaches and formalisms. Here in this section we present a simplified interpretation of this notion adapted to the purposes of data representation and analysis by means of analytical space. Yet being further developed the conception described below can be applied to many other fields. By inference we mean the procedure of finding the mapping between all objects in the source space and objects in the destination space. Thus by means of inference we can for each object in the source space associate one object in the target space. Obviously, it is the definition of function where the source space is the domain and the destination space is the range. In particular, such a mapping can be represented in the form of a table. The problem in finding such a mapping is that source and target subspaces are located in different

parts of analytical space and some formalism is required to calculate the mapping proceeding from the analytical space semantics.

To be able to derive the mapping we need first concrete definition what is the source space (domain of definition) and what is the destination space (range of function). One way consists in choosing concrete elements, which have to be interpreted as containers for domain points and range points. If such elements do not exist then we need to add one element for the source space and another element for the destination space. However for practical purposes building these spaces in explicit form may be not very efficient or even possible. Since we do not actually need explicit representation for carrying out logical inference these space can be defined implicitly. Normally it is done by choosing attribute elements, which constitute the source space multidimensional structure. Additionally, there may be constraints imposed on selected elements.

Inference is based on the process of propagation of information through analytical space. Depending on the direction of the propagation we distinguish deaggregation and aggregation of information. Aggregation means that information moves from lower level child elements to higher level parent elements up to the destination elements. Thus each parent element collects information from a number of child elements and then transfers this information in aggregated form to its parent. The typical form of aggregation operation is sum but there may be other methods including qualitative like creating a set of values and user defined methods. Deaggregation spreads in the opposite direction from higher level elements down to the lower level elements. Normally, information during deaggregation is propagated down to the lowest level. Each child element collects information from all its parents then splits it and transfers down to all its children in deaggregated form.

Inference in analytical space consists of two stages, which are executed for each object in the source domain (Fig. 6). At the first stage a source element is deaggregated down to the lowest level. At the second stage the information from the lowest level is aggregated up to the destination space elements.

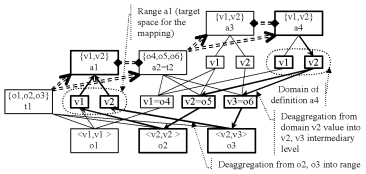

Figure 6. Inference as deaggreagtion and aggregation

In the previous section all elements were defined to have no properties. For practical applications and particularly to easier describe logical inference procedure we should suppose that elements can be described by some primitive property like number, logical value or text string. Concrete type of each element is determined either from context, e.g., from query specification or when the element is created. For example, for numeric attributes all its children are characterised by the corresponding numeric type while for text attributes the elements have strings associated with them. The values associated with elements are directly used in inference by propagating them through analytical space.

To specify concretely how inference is carried out we need to set propagation rules for each element, i.e., what values and how are distributed among parents (aggregation) and children (deaggregation). During deaggregation normally logical rules are used where each element is either enabled (true) or disabled (false). More precisely, if an element if enabled then all its children are also enabled. Such a deaggregation results in a set of objects enabled at the lowest level of analytical space. During aggregation higher level objects collect information from all enabled children and produce one value according to the aggregation function definition. This value is then transferred

upper until the target space is reached.

3 Interactive Data Analysis

3.1 General Procedure

By interactive analysis we mean a capability to view the data from as many different sides as possible. The final goal is to find an interesting view where the data looks simple and informative simultaneously. Different views can be generated both manually and automatically possibly with the appropriate data transformations. This formulation can be opposed to automatic data analysis or data mining where an algorithm searches through the space of hypothesis according to formal interestingness criteria and the search strategy. In interactive analysis it is the space of views and data transformations what is searched for. It is also important that both the search strategy and measure of interestingness are determined by the data analyst from his experience and deeply informal criteria although some stages of this process can be partially automated.

We assume that *the goal of any analysis is simplifying the form of representation by retaining most of the data semantics.* In other words, we want to express most of information in the data in as simple form as possible. Notice that this definition is rather general and, particularly, can be used as a goal of data mining. In interactive analysis the form of representation means choosing data elements to be shown (including structure elements and the data itself) and the type of their representation. It is important that elements of the data structure can be created on demand. In other words, some elements can be deleted (ignored) while new elements can be added into the structure (e.g., new attributes or classifications).

The approach to interactive data analysis described in this section can be described as follows (Fig. 7). The source data is represented in the form of analytical space. This analytical space includes all re-

ally existing data elements and dependencies among them. During the analysis this space is modified by selecting necessarily existing elements and/or creating new elements. This is the most important stage of interactive analysis, which is carried out mostly manually. At the last stage the result of selection is visualised in one or another form, which is chosen from a list of all possible visualisations for the current selection. Choosing visualisation technique means choosing the way elements of analytical space are displayed [4].

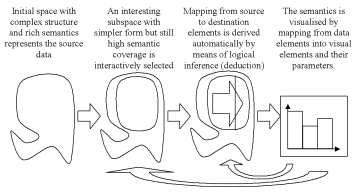

Figure 7. Main stages of interactive data analysis

3.2 Modifying Analytical Space

According to this approach there is a set of all possible (meaningful) analytical spaces, which can be obtained from the initial one by applying a number of elementary modification operations. Thus it is a set of operations that determines the richness and expressiveness of any interactive analysis method. The main modification operations are adding, removing and updating elements of analytical space. Theoretically, these operations are based on and in practice can be reduced to manipulations with element connections (membership relation). In other words, to add new element we just add an isolated entity and

then connect it to other existing elements. To delete an element we just disconnect it from all other elements. Updating means changing the structure of element connections. Such an approach requires clear and deep understanding the analytical space fundamentals and even in this case may be too complicated for real applications. Although in some cases (e.g., fine tuning of analytical space or manipulating really arbitrary and complicated structures) it may be useful we will describe higher level approach where elements are divided into groups according to their purpose. The purpose of this mechanism is similar to that being developed within other formalism for dealing with conceptual structures like Entity-Relationship modelling [7].

Adding elements is the most important operation. To add a *value element* it is necessary to specify at least its container (simple attribute value). Depending on the container it may be required to specify also how this value is characterised in terms of other (orthogonal or hierarchical) containers. This information may be absolutely necessary in the case of the corresponding integrity constraints. It is important that table rows are also considered value objects since they are members of already existing container.

To add a *container* we need to specify its characteristics according to its intended role in the analytical space. Normally for each container we need to know its relationship to other containers, i.e., how it is located within orthogonality and hierarchy relations. The typical cases are adding an orthogonal container for some existing container or adding higher/lower level container. Depending on the added container type its extensional (a set of all possible values) is populated either automatically or manually. For example, for numeric attributes this process is carried out automatically while for finite categorical attributes it is normally manual. Once a container has been added it may be required to characterise objects in terms of its values or connect it in some other way with the rest of the analytical space.

It should be noted that the task of creating and modifying analytical space structure requires certain experience so in many applications the structure already exists and users/analysts have only restricted possibilities in modifying it. For example, it may be possible to change

element parameters.

Deletion of an element is very simple operation from the user point of view but it is more complicated from the formal point of view. The main issue for this operation is maintaining integrity of analytical space by propagating the deletion over the whole structure just like we propagate information during logical inference. This process is also analogous to garbage collection in programming languages. It can be described as follows. The deleted element is disconnected from all other elements. Then all its children are checked for connectivity and if they do not have any parent elements then they are disconnected from their children, i.e., deleted. Then the process proceeds in the directions of new children. It should be noted that this process is propagated from the deleted element down to the bottom level elements where it stops.

Updating elements theoretically means changing the structure of their connections. For example, changing an object attribute value means disconnecting it from the old value and connecting to a new value as a child. In practice it should be implemented as changing element parameters where some parameters can be easily reduced to the element connections while other describe their role in the analytical space and are treaded differently. For example, an element describing an interval of real values has two parameters for its lower and upper bound. Obviously, changing these parameters may influence many other elements in the analytical space explicitly or implicitly.

3.3 Visualisation

Once analytical space has all necessary elements the user can select elements to be included into the source and destination spaces. Given the source and destination spaces the inference procedure automatically builds mapping between them, i.e., for each element from the source space one or more elements from the destination space are found. This information can be easily represented as a table. It should be noted that in general case both spaces may have hierarchical multidimensional structure but in practice their structure should be reasonably restricted

(the goal of analysis is simplifying the form).

The task of visualisation consists in determining how all elements are shown on the screen or other device. The type of visualisation and its parameters have to be chosen in such a way that all data elements and their dependencies are clearly displayed, i.e., we do not loose interesting information discovered on the previous steps. If we have a repository of all possible visualisations with constraints on their use then this can be described as mapping from each element to some visualisation type with additional information how different parameters of visualisation are calculated from element parameters. The problem of finding an appropriate visualisation is not trivial, particularly, currently there exist a great number of them starting from the conventional bar charts and ending with such sophisticated methods as thematic visualisations in geographic information systems [1].

4 Conclusions

The analytical space conception provides highly effective and flexible way of data representation. All data elements are characterised by their position relative to other elements, which is determined by membership relation. All other higher level relations among elements can be derived by means of logical inference where information is propagated through analytical space. On its way down to the lowest level information is deaggregated and aggregated again when it moves up to the target elements. Such an approach to data representation and inference allows us to effectively define interactive data analysis and visualisation procedures. By interactive analysis we mean the procedure of searching through the space of all possible derived data views. The final view should be simple enough on the form and rich on the semantics. The last stage consists in visualising interesting data view by means of available visualisation techniques.

The described approach can be applied to the following problem domain:

- General purpose interactive data analysis tool.
- Data preprocessing tool used to transform data into a form suitable for further analysis, e.g., data mining.
- Data modelling tool used to find data structure suitable for the problem domain.
- Data representation model used to store and retrieve data from databases [5] with analytical capabilities.

This paper focuses on features appropriate mainly for the first task. In future problems specific for other tasks will be considered. In particular, the problem of data modelling and conceptual analysis are of especially great interest so that the results described in the paper can be developed into a new data representation model.

References

[1] N. Andrienko, G. Andrienko, A. Savinov, H. Voss and D. Wettschereck, Exploratory Analysis of Spatial Data Using Interactive Maps and Data Mining, Cartography and Geographic Information Science, Vol. 28, No. 3. July 2001, pp.151–165

[2] A. Berson and S.J. Smith, Data warehousing, data mining, and OLAP, New York, McGraw–Hill, 1997

[3] M. Berthold and D.J. Hand, eds., Intelligent Data Analysis, An Introduction, Springer–Verlag, 1999

[4] S.K. Card, J.D. MacKinlay, B. Shneiderman (eds.), Readings in Information Visualization: Using Vision to Think, Morgan Kaufmann Series in Interactive Technologies, 1999

[5] J.L. Johnson, Database: Models, Languages, Design, Oxford University Press, 1997

[6] A. Savinov, Application of Hierarchical Multidimensional Spaces for Describing Problem Domain Structure, Proc. Conf. On Applied Mathematics & Informatics – AM&I'98, Chisinau, Moldova, August 25-27, 1998 (in Russian)

[7] B. Thalheim, Entity-Relationship Modeling: Foundations of Database Technology, Springer, 2000

Alexandr A. Savinov,
Institute of Mathematics and Informatics,
Academy of Sciences of Moldova
str. Academiei 5,
MD-2028 Chisinau, Moldova

Fraunhofer Institute for
Autonomous Intelligent Systems
Schloss Birlinghoven,
D-53754 Sankt-Augustin, Germany
e–mail: *savinov@ais.fhg.de*

Received December 15, 2001