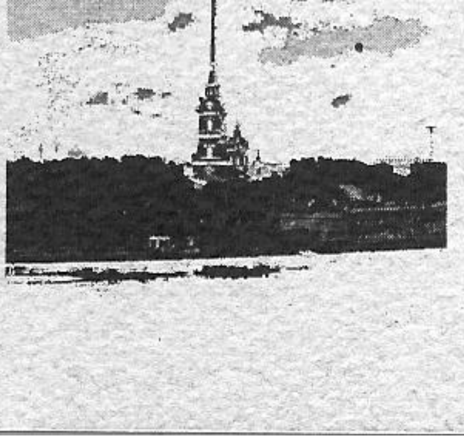
Russian Academy of Sciences
Institute of Problems of Mechanical Engineering
Saint-Petersburg Institute of Informatics

International Scientific School

Modelling and Analysis of Safety, Risk and Quality in Complex Systems



June 18-22, 2001,
Saint-Petersburg, Russia

CONTENTS

KNOWLEDGE REPRESENTATIONS IN INTERACTION TASKS FOR INTELLIGENT PROGRAM SYSTEMS

Adolfo Guzmán Arenas

Center for Computing Research – IPN, MEXICO
aguzman@cic.ipn.mx, aguzman@amiac.org.org.mx

ABSTRACT. Properties of knowledge representations and models are described, in an effort to understand what is common in them, how to link them together, and how to use hybrid representations. Thus, we analyze relational data bases (rdbs), text documents, maps, sounds (audio) and pictures (images).
In addition, some thought is given to the use of hierarchical or tree representations.

I. Reality, Models and Representations

Reality. A given science or discipline studies the properties, relations and transformations of certain things (objects) for a given purpose. These things are thought to belong to the real world. Example: Geology studies the crust of the Earth.... Purpose: to know where it is oil, copper...

Models. Properties and values. Objects studied are described by several properties (measurements, observations) with numeric (numbers, vectors,...) or qualitative (strings, labels, words, concepts) values. Only relevant objects, properties and relations are described. Thus, we make *models* of reality, and store them in a computer, in some *representation* or format. As time passes, models are stored everywhere, some accessible through the Web.

Computer Representation. Most common representations of these models are:

R. Relational data bases (rdbs). An object is represented by a row (record) of a table (file); columns represent properties where values are stored. A relation among objects (such as **a** "works in" **b**) is represented by a table of two columns, named *works in,* where in a row, **a** is stored in the first column, and **b** in the second. Transformations are represented by programs in a language (4GL) associated with the db manager, or in SQL, the Standard Query Language. Rdbs are well described by Codd's Algebra [Codd 70]. Position dictates semantics: 20 in column *weight* is not the same as 20 in column *age.*

T. Text documents. Data is organized in files containing ASCII characters forming words, sentences, paragraphs,... written in a natural language (Spanish, English...). Objects are represented by nouns; relations or processes, by verbs. The semantic of the document is given by the associated language (Spanish, say).

M. Maps. Objects have position in the Earth or in the Universe. A standard coordinate system (latitude, longitude, elevation, or ϕ, λ, ρ) there exists. There are four types of objects: <u>point</u> objects (a gold mine, say), represented by a rdb row where, in addition to its properties (quantity of gold produced per year, owner...), ϕ and λ are stored; <u>lines</u> (a river, say), represented by a row having associated a set of points ϕ_1, λ_1, ϕ_2, λ_2, ...; <u>areas</u> (a lake, say), represented by rows having associated a set of points ϕ_1, λ_1, ϕ_2, λ_2, ... representing its boundaries; and <u>volumes</u> (an oil field underground), typically represented by a collection of boundaries, each at different depth. Relations ("to the north of", "20 kilometers from", "surrounded by", "inside"...) are not explicitly stored, but they are derived (computed) from the coordinates of the objects.

S. Sounds and signals of the form y = f(t) are represented by an array (vector) of numbers. Relations are not explicitly stored.

I. Images and data of the form y = f(x, y) are represented by a 2-D matrix of pixels, each of them is a vector (to represent color). Relations are not explicitly stored.

V. Videos. Similarly, *videos* are sequences of images.

Mathematical, Music and other representations are not studied, only those above.

II. Hierarchies and Invariants

Most times, objects in real life form structures with levels, called hierarchies.

Hierarchies of objects. Most objects form hierarchies: Sandal, moccasin, boot, are subsets of shoe. Shoe, shirt, sock, pant, are subsets of wearing apparel... Continents are divided into countries, these into states, these into... The most common relation to form hierarchies is "subset". Ontologies or trees of concepts are other names for hierarchies. We refer to: seeing things at a higher level or in more general form or in less details or at smaller scales.

Hierarchies of Relations abound. Thus, "last son of" is a specialization (subset) of "son of", which is a specialization of "parent of", which is a specialization of "relative of".

Hierarchies of Transformations are common, too: *to limp* is a certain type, a kind (a subset) of *to walk*, which is a kind of *to move on the surface*, which is a kind of *to change position* (to move), which is a kind of *to change* [Lenat & Guha 94].

Hierarchies of Purposes. Purposes are also structured and form hierarchies.

It makes sense to use, therefore, a hierarchical representation for our models, and then to use the same programs at different levels for the same kind of processing. For this reason, the representation is some times called *recursive representation.*

Invariants. The "useful" or important *transformations* on a model align with (respond to) some *purpose*, and must yield approximately the same value, even when details or precision of representation vary. For instance, the answer "yes" to the question or transformation "Is lake Taho inside U.S.?" must not change with the scale of the map. These unchanged values are called **invariants** of the representation, akin to eigenvalues. Example: computing the length of the Amazon River must yield roughly the same value, no matter the map scale. Example: the *main topics* of an article [Guzman 98] must be the same, even if the document is translated to French, or abridged. Example: the answer to a question "Is this Adolfo Guzman?" to a given picture P, must be the same irrespective of image resolution.

Energy. In a hierarchic representation, high levels must represent most of the useful information, although most of the information (details, kinks of a river, specs in a face) resides in the lower levels. For lack of a better name, we call **energy** this "useful information."

III. Mappings a representation into another, and mixed processing

We refer to representations R, T, M, S, I already described in section I.

1. **From rdbs (R) to geographic data bases (M), and back.**

handled by: add to that object from R a geo position (φ, λ, ρ).

Relation mapping. Relations from R (such as *owns*) usually do not have geographic meaning. Relations from M (such as *to the north of*) can be computed and stored in a table of A but, since the relations in a map are many, this is not practical.

Mixed processing. To mix queries to R with queries to M (geographic, position, distance... questions), one must implement the geographic predicates [which work on φ, λ, ρ] and mix them at run time. Thus, the question "give me all the people with salary > 3,500 and living north of Quebec" consists of the A predicate "salary > 3,500" (which will be answered by looking at column "salary") and the predicate "north of", which must be computed at run time. See *Mixed processing* in (3).

2. **From signals (S) to rdbs (R).** See (3) next.
3. **From images (I) to rdbs (R), and back.** Mapping a R object or value (the weight of A.Guzman) to a S or I object or value does not make sense, in general.

   Object mapping. An object f(x, y) of I is mapped to a relational table ***f*** of R whose first column contains x·y (x concatenated by y; this forms the *key* of table *f*), its second column containing the value of f. More often, rdbs are *extended* to store "binary large objects", or BLOBs, that are handled through their own functions (methods). Thus, a picture is stored as a BLOB, and the db software (or the user) provides functions such as value_of(x, y) to extract the value of f at (x, y).

   Relation mapping. S, I relations are not explicitly stored in R.

   Mixed processing. In the Informix blades, the user provides predicates (for instance, "sings rhythmically", applied to the song of a bird in a BLOB) and functions (such as "percentage of quiet periods in song"). These are mixed with normal R statements, and are handled by the rdb manager. Example: give me all birds that weight less than 100 grams (a R relation) and that sing rhythmically (a S relation). User predicates execute at run time; their value is not pre-stored in R.

4. **From text (T) to rdbs (R) and back.**

   Object and relation mapping. Objects can be mapped from R to T by mechanically generating text sentences such as "A.Guzman's *age* is 50", "A.Guzman ***lives in*** Mexico City"... from tables in R *age* and *lives in*. To map T objects to R, a text program (containing a parser) must find all nouns or substantives, such as A.Guzman, as well as its properties (weights 75 Kg.) and relations (lives in Mexico City), and store them into tables in R. The approach has to handle synonyms and more general objects, since the text may say that A.Guzman *lives in* Mexico City, while B.Ferro *dwells in* Mexico,D.F. The trees of Section II can be used, but we prefer to map only the most popular (most often asked, most useful) objects and relations, into a few tables of R that can be seen as a "summary in R" or "Statistics in R" of text T. Thus, often asked objects and relations are pre-stored (or cached) in R; the rest are not.

   Mixed processing. We follow the same approach of (3) preceding: to treat text as BLOBs of R with their own methods, and to handle with SQL the few tables of the summary in R of text T. Nevertheless, this approach is deficient, because the summary in R is small [although its energy is large], and the handling of T with its own methods is slow (they can not pre-store values; they run at query time).

When to use R? When relations are few and their use is well known. Values can be text.

When to use T? When relations are " grammatical" or "Spanish". When users often seek unexpected or unfamiliar relations, so that predefined relations (tables) will not suffice.
When to use M? When data naturally has position in space.
When to use S, I, V? Only when the model is formed by sounds, images or videos.

Hybrid or unified representations (and data bases). Low costs of processing and networks enable a person to have access to large quantities of data stored in several manners. There is an urgent need for mixed processing and for mixed representations.

- Businesses seek (buy, use, and return) "Knowledge management" systems.
- Most data is on the Web, in representations R, T, M, S, I, V.
- For modern Integrated Information Systems, such as those that handle scientific, technical and project data, for a large organization; National Video Libraries; easy Data Locator in a large University or organization; a Center for Engineering Studies and Development; Regional Planning; agencies that mix map data with statistic or census data.
- Current Integral Information Systems are all relational. Example: SAP. Their failure, in many cases, to represent complex, country-wide models (i. e., science activities in Mexico) shows the limitations of rdb technology. "No matter what relations you build, the user wants to see something else". The approach of rdbs to pre-store in rigid tables every object and relation that can participate in a query, has shown its limits.

Computer-aided mixed processing. We are designing a system that can handle models represented in R, T, M, S, I, or V, but has the user in the loop.

i. To the representation of each model in the mix, add a summary in R [By computer, as in (4) above, or by hand], or a summary in T (an English description; key words), or both, or a summary in R where some columns have text as their values [Example: Summary in R of a Project Proposal in T has columns "Author", "Field of Study", "amount"...].

ii. Use the summaries plus the user in the loop, to collect and present models that could be useful for the study. For summary processing, use the techniques for R+T handling of (4). This stage is an "information gatherer" or "information enricher" that collects relevant models, in spite of its representation.

iii. Now, use the natural methods for the representation at hand: if model k had the representation $\{M + R_{summary}\}$, it is time to start using M (map) primitives, predicates, etc.

iv. Mixing information from several models (for instance, combining data from a table with data from tables in other dbs, or in a map) can only be done with the user in the loop.

References

Codd, Edgard. *ACM Turing Award Lectures: The First Twenty Years*, Addison-Wesley, Massachusetts, 1987.

Douglas B. Lenat and R. V. Guha. *Building large knowledge-based systems.* Addison Wesley. 1990

Guzmán, A. Finding the main themes in a Spanish document. (1998) *Journal Expert Systems with Applications*, Vol. **14**, No. 1/2, 139-148, Jan./Feb.