Computational Mechanisms for Knowledge Organization

Abstract. This paper reviews several knowledge organization techniques used in Computer Science, in areas such as Artificial Intelligence, Databases and Software Engineering. Some of these computational mechanisms may assist in the organization and management of immense digital information resources. At the same time, the paper notes an increasing need for computer-based information systems to operate in open networked environments. This need requires knowledge organization principles, which are flexible and can be used with informally expressed knowledge. We expect to find such knowledge organization techniques in Library and Information Sciences, and hope to integrate them with the computational techniques described in this paper.

1. Introduction

Software systems routinely handle knowledge about the world. For example, a software system may store data in a database, which represent facts, such as peoples' age, address and telephone number. In addition, the system may have knowledge embedded in its code, such as «a person's age is usually an integer between 0 and 120», which is used to validate incoming data.

Over the years, different areas within computer science have developed techniques for representing and organizing knowledge so that it can be accessed and used by humans and software systems alike. In particular, Artificial Intelligence (AI) has developed techniques that make it possible to represent knowledge so that it can be exploited by «intelligent» systems. Databases and Information Systems have focused on techniques, which lead to high level, «conceptual» descriptions of the contents of a database. Software Engineering has developed elaborate techniques for capturing requirements, design decisions and rationale for a software system.

This paper reviews some commonly used knowledge organization techniques in computer science. As more and more information resources become available in digital and electronic form, some of these computational mechanisms for knowledge organization may be used to offer greater support for integrating, organizing and managing the vast amounts of information. At the same time, the increasing need for systems to operate in open networked environments calls for more flexible knowledge organization techniques in computer systems, such as those used in the library and information sciences.

2. Conceptual Modeling

In Computer Science, conceptual modeling is concerned with the construction of computer-based symbol structures (hereafter information bases) which model some part of the real world (or, application). A conceptual model comprises a collection of:
1. Primitive terms intended to describe an application;
2. Abstraction mechanisms for organizing an information base;
3. Primitive operations, which can be used to update and access an information base;
4. General integrity rules, which define the set of allowable information base states, or changes of states.
Over the years, many conceptual models have been proposed. Semantic networks represent directed, labeled graphs for modeling the structure of human memory (Quillian, 1968), where nodes represent concepts. The semantic model for databases (Abrial, 1974) and entity-relationship model (Chen, 1976) were intended as advances over logical data models, such as Codd's relational model.

The entity-relationship model is an excellent example of a conceptual model. Its primitive terms are those of entity, relationship and attribute. The model doesn't offer any abstraction mechanisms. Its associated operations include add, remove, and update operations for entities and relationships. Finally, the model supports cardinality integrity rules, such as every child has up to two parents.

Another important example of an early conceptual model is the Structured Analysis and Design Technique (SADT™) which was proposed as a «language for communicating ideas» (Ross, 1977). According to SADT, the world consists of activities and data. Each activity consumes some data, represented through input arrows from left to right, produces some data, represented through output arrows from left to right, and also has some data that control the execution of the activity but are neither consumed nor produced. For instance, the Buy Supplies activity of figure 1 has input arrow Farm Supplies, output arrows Fertilizer and Seeds and control arrows Prices and Plan & Budget. Each activity may be defined through a diagram such as that shown in figure 1 in terms of sub-activities. Thus Growing Vegetables is defined in terms of the sub-activities Buy Supplies, Cultivate, Pick Produce and Extract Seeds.

3. Abstraction Mechanisms

By definition, abstraction involves suppression of (irrelevant) detail. For example, the generic concept of employee might be abstracted from those of secretary, teacher, manager and clerk by suppressing particular features of these concepts (teachers teach a subject, managers manage some group of people), and focus on commonalities (all employees have a salary, a position, a job description, etc.).

Abstraction mechanisms organize the information base and guide its use, making it easier to update and search it. Hence, abstraction mechanisms have been used in computer science even before the advent of conceptual models (e.g., ALGOL 60 and LISP). Below we
discuss a number of abstraction techniques that have been supported by conceptual models.

3.1. Classification

Classification (Motschnig, 1992) is fundamental for human cognition, conceptual models, and information bases. Here, a unit (e.g., entity, relationship, attribute, activity) within an information base is classified under one or more generic units (classes). Instances of a class share properties, e.g., all units classified under the class Person have an address and an age attribute, while others classified under Dog, can participate in a master relationship and are constrained to have exactly four legs. Some models support recursive classification, i.e., classes may (or must) themselves be instances of other classes. In this case the class Person might be an instance of the (meta)class AnimateClass, which has as instances all classes describing animate entities.

Classification has been used to support syntactic and semantic consistency. For example, sorts in mathematical logic and types in programming languages are used mostly for syntactic checking. So are tables or relations in the relational model for databases. In semantic networks and object-oriented information models, classification distinguishes between tokens or objects, which represent particular individuals in the application, and types or classes, which represent generic concepts.

Classification has been studied in the machine-learning community to support information clustering (using inter-object similarity) (Cheeseman, 1988, Stepp, 1986), and problem solving (using previous similar experience) (Turney, 1995). Knowledge-based or conceptual clustering employs feature similarity to organize knowledge in knowledge bases (Mineau, 1992) or libraries (Murty, 1995). In a conventional data analysis, objects are clustered into classes based on a distance (similarity) measure. The similarity of two objects is represented as a number – the value of a similarity function applied to symbolic descriptions of objects. Thus, the similarity measure is context free. Such methods fail to capture the properties of a cluster as a whole and are not derivable from properties of individual entities (Stepp, 1986). A context-based classification helps to retrieve all entities that are relevant to a given task, which improves a system's accuracy and flexibility (Jurisica, 1997).

3.2. Generalization

Generalization (Brachman, 1983) organizes generic concepts in an information base into taxonomies, referred to as isA or generalization hierarchies. For example, the class GradStudent may be declared as a specialization of Student («Every grad student is a student»), which in turn a specialization of Person («Every student is a person»).

Inheritance is a fundamental ingredient of generalization hierarchies. Inheritance is an inference rule that states that attributes and properties of a class are also attributes and properties of its specializations, or isA descendants. Thus, Student, and transitively GradStudent, inherit the address and age attributes of Person. Such inheritance may be strict in the sense that constraints on attributes and properties can be strengthened for more specialized classes but cannot be overridden, or it can be default in the sense that overriding is allowed. For example, given that the values of the age attribute for the class Person can range from 0 to 120, no subclass of Person is allowed to have an age attribute with values outside the range 0–120 if strict inheritance is enforced. With default inheritance, on the other hand, one can define a subclass MythicalPerson where the age attribute is unbound.
3.3. Aggregation

Aggregation (Motschnig, 1993) views objects as aggregates of their components or parts. Thus, a person can be viewed as a (physical) aggregate of a set of body parts -- arms, legs, head and the like -- or as a (social) aggregate of a name, address, social insurance number, etc. Components of an object might themselves be aggregates of other simpler components.

There is solid psychological evidence that most of the information associated with a concept is of the aggregation variety (Miller, 1976). Within Computer Science, (Kim, 1989) proposed a formalization of aggregation within his object-oriented data model in order to move away from pointer-type references between objects. In his proposal, components may be dependent on the aggregates to which they belong. This means that if an aggregate is removed from the information base, so are its dependent components. Likewise, a component may be exclusive, which means that it can only be part of a single aggregate. In addition, aggregation may be strictly hierarchical or recursive. For instance, an employee may be defined as the aggregation of a department, a salary and another employee who serves as the employee's manager. Finally, an object in the information base may be treated an aggregate in more than one way. For example, an organization may be described as a hierarchical aggregation of different managerial levels (managerial perspective), or as a vertical aggregation of departments serving different functions, such as production and marketing (administrative perspective). Figure 2 uses a particular notation for aggregation to describe organization as an aggregation of different departments (finance, production, etc.), as well as an aggregation of different managerial levels.

![Figure 2: Multiple decompositions of the concept of organization](image)

3.4. Contextualization

Contextualization offers mechanisms for partitioning of the information base into multiple, possibly contradictory, viewpoints on the same objects, both during modeling and during use. Reasoning with information in context has a positive effect on classification accuracy and system flexibility (Jurisica, 1997).

Various forms of a contextualization mechanism have been used in advanced information systems (Norrie, 1994). Programming language modules, scopes and scope rules determine which parts of a program state are visible to a particular program segment. Since the early days of AI, contexts have found uses in problem solving, as means for representing intermediate states during search for a solution (Hewitt, 1971), in knowledge representation,
as representational devices for partitioning a knowledge base (Hendrix, 1979). In CAD and Software Engineering, *workspaces, versions and configurations* (Katz, 1990) are generally accepted notions offering respectively mechanisms for focusing attention, defining system versions and means for defining compatible system components. Database *views* have been traditionally used to present partial, but consistent, viewpoints of the contents of a database to different user groups.

### 3.5. Materialization

Materialization relates a class to a more concrete class. For example, a (theatrical) play, say «Hamlet», is related to particular productions of the play, say the one now playing at the Royal Alexandra Theater. These can be further materialized by particular shows of each production, such as the matinee show on October 26, 1997. This is clearly a useful abstraction mechanism for manufacturing applications, which involve multiple, often indistinguishable entities, of the same type. As argued in (Pirotte, 1994), the formal properties of materialization constitute a combination of those of classification and generalization.

### 3.6. Normalization

Normalization (Borgida, 1985) models typical entities first and treats exceptional situations afterwards. Naturally, there must be some systematic way to find the abnormal cases and a way to specify the exceptional circumstances as annotations that do not interfere with the typical entities. Because generalization may lead to over-abstraction there must be a mechanism to analyze and deal with the conflicts (Borgida, 1988).

### 3.7. Parameterization

Parameterization is a mathematical abstraction technique that has been used with great success in programming and formal specification languages such as OBJ and Z. Among requirements modeling languages, ERAE and its successors (Dubois, 1992) support parameterization to enhance the reusability of requirements. For example, one may define a requirement model with two parameters *resource* and *consumer*, which includes actions such as request and grant and constraints such as «a grant will take place for an available resource if there is a waiting consumer». This model can then be instantiated with parameter *resource* bound to *book* and *consumer* bound to *libraryUser*. This abstraction mechanism is also a vital part of analogical reasoning systems (Carbonell, 1981).

### 4. The Evolving Needs for Knowledge Organization

Knowledge organization has played an important role in the evolution of software systems. New mechanisms have been introduced at various times during the short history of Computer Science to meet changing needs. In comparing the main abstraction mechanisms reviewed in this paper with those in Information Sciences, one may find that the computing discipline has focused mostly on those that can be specified precisely, and which therefore can be formalized. This is not surprising since the field is concerned with the management of knowledge with the intent that at least some part of that management will be done computationally. The result is that these knowledge organization schemes can only deal with restricted classes of knowledge, namely those that have been sufficiently formalized to admit abstractions of the kinds described above. This distinctive approach has served the computing field well up until recently.

Today, with the advent of Internet technologies and emerging global information infrastructures, the potential beneficiaries of the knowledge encoded in a particular
information source, a database or text file, is not confined to the initially intended users, and may potentially include the whole world. Conversely, one can substantially amplify the utility of any one particular information system by enabling it to draw on worldwide information resources. The current work on «ontologies» in Computer Science recognizes the need for computer based information systems to share information (Vickery, 1997).

It follows from these trends that computer-based information systems can’t be treated any more as closed environments managing prefixed types of information used by a small and coherent group of users. Instead, such systems need to become open in that they can access and offer to their users any information that is relevant to the application, as soon as it becomes available. The Library and Information Sciences have a long history of scholarship, theory and experience in dealing with information and knowledge from this broader perspective. We therefore anticipate that knowledge organization concepts and principles from these disciplines will have a much greater impact on software systems in the future. Among others, this impact will lead to abstractions, which allow ambiguity, fluidity, flexibility, and human participation and control in the organization of knowledge.

For example, a great proportion of human knowledge is expressed in natural language, so that string- or thesaurus–based mechanisms will be crucial (Williamson, 1996). Computational techniques such as information retrieval (Jones, 1997), data mining, knowledge discovery (Jurisica, 1998), and semantic analysis (Soergel, 1996) offer complementary support. Such techniques may support what Hjörland calls pragmatic and scientific classification (Hjörland, 1994).

Organizing information on the Internet is a challenging task because it requires us to deal with immense digital information resources, which are unstructured, distributed, and dynamic. Several automatic and semiautomatic techniques have been designed to help with this process. Schwartz (1997) proposed an Internet indexing standard X-DEX based on the US Library of Congress cross-reference system. The success of Yahoo! is another example of the power of combining library and information science techniques with computational techniques. Web pages located by Yahoo’s Web spider and those submitted by page owners are classified by cataloguers who design and populate Yahoo’s catalogue (Brisson, 1997). The number of visits the site receives daily attests to the success of this approach. However, much remains to be done if we are to lay good scientific foundations for the complementary use of knowledge organization techniques from Computer Science as well as Library and Information Sciences. As stated by Hjörland «Any given categorization should reflect the purpose of that categorization. We must choose the purposes of our classification.» (Hjörland, 1994).

References


Advances in Knowledge Organization, Vol.6(1998)


