Journal of the University of Chemical Technology and Metallurgy, 47, 2, 2012, 230-234

TOOLS FOR A UNIFORM APPROACH TO SUBJECT AREA STRUCTURING IN MACHINE LEARNING

V. Nikolov, S. Panov, E. Yanev

University of Chemical Technology and Metallurgy
8 Kliment Ohridsky, 1756 Sofia, Bulgaria
E-mail: nikolov_vnz@abv.bg,
smpanov@yahoo.com, e_janev@abv.bg

Received 02 December 2011
Accepted 12 March 2012

ABSTRACT

Two basic types of relationships applied on a class-object model from Object Oriented Programming that can efficiently describe a variety of subject areas (SA) are presented. SA can be depicted with growing complexity by hierarchical layers. The importance of AND/OR logical operations to control transactions between objects is discussed. Main operations on classes and objects are outlined.

Keywords: hierarchical layer, subject area, AND/ OR logical operations, machine learning system.

INTRODUCTION

The development of cognitive sciences and artificial intelligence in the last decade led to the development of machine learning systems which demonstrate remarkable intelligence when conducting certain machine learning processes. The vast scope in which the machine learning systems are used calls for the intensification of this kind of software production, and the requirement for intelligent and user-adapting machine learning call for the development of a uniform approach to the relationships and interactions of the objects describing this SA.

At present, the majority of the existing machine learning systems focus on the contents of the material, at the expense of the structure and presentation of knowledge, in order to improve learners' abilities to apply the knowledge for solving different application tasks. Significantly, some of the structures used for presenting knowledge in a particular SA are more appropriate than those used in another SA – something that should be taken into consideration when designing development tools.

REQUIREMENTS

Taking into consideration that the structuring of knowledge (SA structuring) significantly influences the efficiency of the machine learning process, the following requirements to the functional possibilities of the programming tools can be formulated:

- Possibilities for the integration of new knowledge or exclusion of already existing knowledge from what has already been described in the particular SA.
- Flexible alteration of the relationships between the objects describing the programming units, in order to improve the knowledge structure.
- The tool environment is required to create levels of evolutionary complexity which represent the relationships between the objects in the SA. This requirement is imposed by both the necessity for expanding the basic knowledge and the very hierarchical essence of knowledge.
- Because machine learning is a two-sided process – generalization and detailing – the generated structure of the SA also needs to take into consideration these two peculiarities [1].
- The definition of classes must provide the opportunity for maximum versatility when describing the SA.

AN UNIFORM APPROACH TO THE RELATIONSHIPS IN THE SA

From the machine learning perspective, each processing unit is regarded as a goal that has to be reached [1]. Again, in the light of this terminology, the

relationships between the different objectives will be discussed, as well as the SA objects. The relationships we will discuss are of two types: **COMPOSE-OF** and **A KIND-OF.** Actually, those are the relationships between the classes through which the uniform approach of the SA will be applied, and the relationships of belonging and inheritance within it will be modeled. In order to detail and generalize the SA knowledge, we will introduce the concept of "hierarchical layer" [1]. A hierarchical layer is hereby defined as a set of semantically related objects which determine the detailing level of knowledge.

Firstly, each class will be related to a function which puts in congruence the set of object names of the respective class. If those class names are related to a particular level of SA detailing, then:

$F_j (C_i (OB_{ij}))$,
where:
- F is the function which is related to searching in the hierarchical layer *j* and which puts in agreement the set of classes included at that stage of knowledge detailing;
- C is a function producing a set of objects related to the respective class *i* which belongs to the layer *j*;
- OBij – are the objects obtained from the above presented two functions. These objects are located in the detailing level *j* and belong to class *i*.

Fig. 1. illustrates the relationships between the layers, classes and objects within the framework of a single SA. As a result of the application of those functions, all objects belonging to the respective layer of the SA are produced. The most convenient way to generate the set of those objects is to apply recursive search on the graph of the SA [3].

Each of the objects that need to be generated shall be regarded as:

1. Comprised of other objects which may belong to different classes.

2. As an object of a class which inherits the properties of other classes.

The concepts of class hierarchy and nested classes are presented in [5].

The first type of relationships suggests that the relationship **COMPOSE-OF** is imposed over the set of object names, thus demonstrating that one object can be presented as a simple set of other objects, obtained through the operation of inclusion. An object named OBi will be called "a simple object" if the class producing it is not a

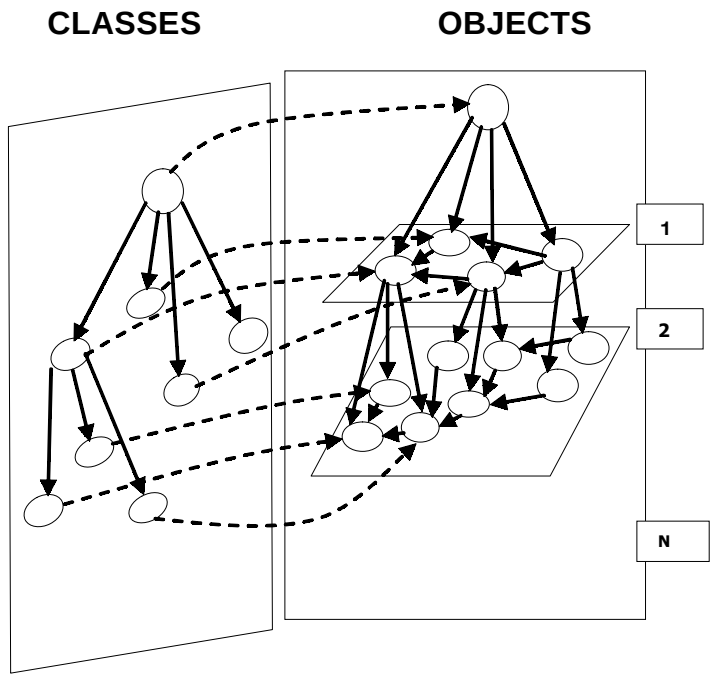

Fig 1. Class and object diagrams in the subject area (SA).

result of the relationship **COMPOSE-OF**, i.e. we have a base class. Although this could be allowed:

C2 **COMPOSE-OF** (C1),
it would lead to 2 equivalent classes which is meaningless. The participation of a single object in the creation of a complex object is meaningless also because its properties may be included in the inheritance part of its parent class.

The relationship **COMPOSE-OF** over the set number of classes may be interpreted as follows. If C1, C2 . . . Cn are classes fulfilling the condition:

C **COMPOSE-OF** (C1, C2, ... Cn),
then the object OB of the class C is composed of objects which respectively come from the classes C1, C2, . . . Cn. The set of classes must include at least 2 classes. This relationship between the objects may be defined as a "relationship of inclusion." It is characterized by the absence of such strong dependence between the different classes (respectively, objects) which exists in the relationship of "class inheritance". In that relationship, the objects do not exchange information between their interface sections.

Another type of relationship on the SA meta-level is the class-subclass relationship **A-KIND-OF**. It determines the structure of the class hierarchy, i.e. the relationships of inheritance between the different classes which represent the SA at the conceptual level. If we mark by C the set of classes (C_i being one of those classes), then $C_i \in C$ is true, and the relationship:

Ca_i **A-KIND-OF** Cp

will demonstrate that Cai is the *i*-number successor of the parent Cp. In the event of simple SA inheritance, the parent is only one. However, by multi-way inheritance, the properties of a couple of objects are inherited simultaneously. Then the following uniform approach will be applied:

$$Ca_i \; \mathbf{A-KIND-OF} \bigcup_{j=1\ (i \neq j)}^{n} Cp_j$$

where Cp is the set of parents of the successor Ca_i.

Taking into consideration the given interpretation of the relationships **COMPOSE-OF** and A **KIND-OF,** their basic properties may formally be presented as follows:

1. C-class _____ By definition, C is a class.
C **A-KIND-OF** C

2. C1 **A-KIND-OF** C2, C2 **A-KIND-OF** C3
C1 **A-KIND-OF** C3

If class C1 inherits C2, and class C2 inherits C3, then class C1 also inherits the properties of class C3.

3. C1 **A-KIND-OF** C2, C2 **A-KIND-OF** C1
C1 **A-KIND-OF** C2

If two classes mutually inherit the properties of each other, then they are equivalent.

4. If the class C_i is comprised of the combination of the base classes Cj, then it can be presented as follows:

$$C_i = \bigcup_{j=1\ (i \neq j)}^{n} C_j$$

The restriction $j \neq i$ prohibits the recursive inclusions between classes in the subject area.

AND/OR LOGICAL OPERATIONS IN THE MACHINE LEARNING ENVIRONMENT

To formalize the notion of depth of architecture, one must introduce the notion of a set of computational elements [8]. In our case, except for the relationships of inclusion and inheritance, in the SA development two new types of relationships need to be included, namely the conjunctive and disjunctive relationships between the objects describing it.

- All machine learning influences of the given object set have to be encompassed for the acquisition of new knowledge and skills, then those objects will be united through conjunctive relationships at the respective level of the SA hierarchical abstraction.

- Provided that the acquisition of new skills requires the encompassing of several machine learning influences defined above a certain object set, then we have an example of disjunctive unification of those machine learning influences.

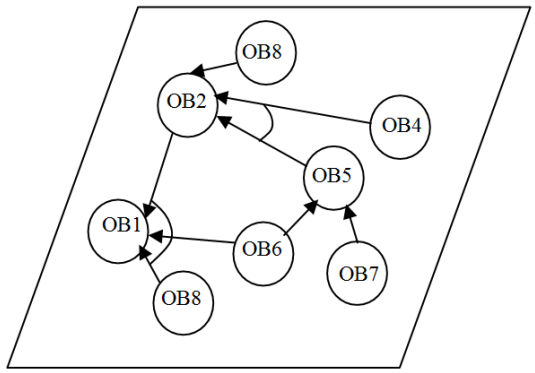

Fig. 2. Directed AND-OR graph in the (SA).

The logic of the relationships between the different objects is determined entirely by the SA semantics, as well as by the expert who defines it. Those types of relationships are allowed only within a single level of abstraction of the machine learning environment and will determine the navigation in the learning material. In this context, a layer of the SA hierarchy could be presented as directed AND-OR graph (Fig. 2). In this particular case, OB1 may be regarded as an object or a learning unit to be acquired, which requires the conjunctive interaction of the objects OB2, OB3 and OB6. The conjunction is represented in the form of arrows joint together through an arc. In order to acquire the knowledge related to the object OB5, the disjunctive interaction between the objects OB6 and OB7 must be applied, while the knowledge corresponding to the object OB2 presents disjunction between the object OB8 and the conjunction of the objects OB4 and OB5.

In the described SA certain objects don't have any parent objects. These objects define the set of base units of a knowledge level. These units should be derived at the structuring stage of the learning material. In the particular case those are OB3, OB4, OB6, OB7 and OB8. For those objects, the following relationships are applied:

1. Provided that Di and Ki are sets of objects, which are respectively in a conjunctive and disjunctive relationship regarding Ob_i for the particular layer, then:

$D_i \cap K_i = \varnothing$

This means that these sets have no common objects.

2. The Obi objects for which the following relationships are true:

$(D_i = \varnothing) \Lambda (K_i = \varnothing)$

represent the basic knowledge of the hierarchical level which is the subject of a uniform approach. This means that no disjunctive or conjunctive relationships lead to them.

3. Provided that Di and Kj are the sets of objects defining conjunctive and disjunctive relationships in the object Obi, then the following is true:

$(OB_i \notin D_i) \Lambda (OB_i \notin K_j)$

This practically means that cycles in a graph developing the respective level of SA detailing, are not admissible.

A hierarchy is developed through the tool environment, and at the top of this hierarchy lies an amorphous class R, i.e. a class for which the operation C_i **COMPOSE-OF** R is invalid (because we need more than one class on the right hand side of the relationship). Only the operation C_i **A-KIND-OF** R can be performed on it.

The implementation of the relationships between the classes and their instances is a matter of expert decision by the one who develops the SA.

OPERATIONS ABOVE CLASSES AND OBJECTS

The eligible operations above the classes and objects are selected so that they allow the implementation of the SA semantics. Let us assume that C is a set of all primary classes – including the primary classes and those derived from the base ones by applying KIND-OF on one class only. From the classes involved therein through the operations of inclusion and inheritance, new derived classes can be obtained.

1. Development of new classes from the existing base ones. By using the relationship **COMPOSE-OF** a new class Ck can be obtained through the union of a part of the classes C_i which belong to C.

Let us assume that class C1 is different from class C2. We will discuss the relationships in which a newly obtained class Ck would find itself regarding those two classes from which it was derived through the basic operations of $\cup$ and $\cap$. Those relationships also spread over the rest of the derivative classes:

Ck **COMPOSE-OF** (C1 $\cap$ C2) V Cк **COMPOSE-OF** (C1 $\cup$ C2)

This means that the class Ck is either comprised of the objects which are common both for the classes

C1 and C2, or it is a result of the uniting of their respective constituent objects. If the objects of C1 and C2 contain simple objects which are common for both classes, the same objects are found only once in Ck.

2. When developing new classes, another possible operation is the exclusion of a particular class from a class in which it participates. This operation is directly related to the one of excluding objects defined for this class. If the class C_i consists of a composition of the base classes, then it can be presented as:

$$C_i = \bigcup_{j=1\ (i \neq j)}^{n} C_j$$

Provided that the operation ***name*** *(Ck)* finds the name of the class Ck, then the operation ***del*** *(Ck)* will exclude the class Ck from the derivative class and the result will be as:

$$C_i = \bigcup_{j=1\ (i \neq j)}^{n} C_j \qquad \text{where } j \neq k \ \wedge i \neq k.$$

The next activity to be performed is the search within the SA, developed by the instances, in order to exclude the objects which have been identified with the already excluded class.

3. Operation of display of all classes involved in the composition of a particular class.

4. Another function required to ensure the proper SA semantic representation is checking if a class exists within another developed class. As a result of the function execution, a Boolean variable is obtained:

if $C_j \in C_k$ then Val = true

if $C_j \notin C_k$ then Val = false

CONCLUSIONS

On the grounds of the proposed data structures, methods servicing the interface part of the classes, used for SA descriptions, are developed. Furthermore, machine learning strategies, which determine the navigation within the machine learning environment and their implementation in the system, should be developed, and functions for developing a learner's individual model should be created.

REFERENCES

1. P. Karampiperis, D. Sampson, Adaptive Learning Resources Sequencing in Educational Hypermedia Systems, Educational Technology & Society, **8,** 4, 2005, 128-147.
2. S. Russel, P. Norving, Artificial Intelligence a modern approach, Third Edition, Prentice Hall, ISBN-13: 978-0-13-207148-2, New Jersey, 2010, pp. 1132.
3. N. Nilson, The Quest for Artificial Intelligence, Stanford University, USA, 2010, pp. 707, ISBN: 9780521122931.
4. N. Nilson, Introduction to Machine Learning, Stanford University, USA, 2005, pp. 179.
5. A. Willms, Einstieg in Visual C++ 2008, ISBN: 978-3-8362-1193-2, Galileo Press, Bonn, 2008, pp. 642.
6. Herb Shildt, C++ Programming Cookbook, pp. 495, 2008, McGraw-Hill, ISBN-13: 978-0071488600.
7. R. Michalski, J. Carbonell, T. Mitchell, Machine Learning an AI Approach, Vol. II, Morgan Kaufman Pb., 1986, California, pp. 738.
8. Yoshua Bengio (2009). Learning Deep Architectures for AI. Now Publishers Inc. p. 14, ISBN 9781601982940.