A note on the representation and the manipulation of structures

Amedeo Napoli
LORIA – UMR 7503
B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France
(Email: napoli@loria.fr)

1 Introduction

In this position paper for the DL'98 workshop, we present a discussion on the notion of structure and on the related reasoning problems. Structures are widely used in artificial intelligence to represent real-world concepts, especially in design, configuration and classification problems. A structure is a composite object – composite objects and structures are strongly related– which has a number of parts that are interconnected and that can be composite objects themselves (structure = collection of parts + configuration information). Formally, a structure can be defined as a set of instantiated relations. By contrast, a graph is a particular kind of structure representing only one binary relation.

In the following, we discuss three approaches used to represent and manipulate structures: structures as composite objects in frame-based representation systems, structures as composite descriptions in description logics (complete discussions on composition may be found in [Artale *et al.*,1996] and in [Lambrix,1996]), and finally structures as structural descriptions in the field of pattern recognition. We briefly present these three approaches and then we discuss their connections and open problems.

2 Structures and composite objects in a frame-based representation system

In a frame-based representation system, a real-world concept is represented as a class or a frame composed of a collection of slots denoting the properties of the concept [Napoli *et al.*,1994]. The slots are divided into attributes and methods, describing respectively the characteristics and the behavior of the concept.

A structure may be represented as a composite object –"composite frames" or "composite individuals"– in a frame-based representation system. A composite object is an aggregation of components, each component being an object and describing a part of the composite object. It may be organized into a composition hierarchy, where the (name of the) composite object is at the root of the hierarchy, while its components are in the immediate lower level. The components may be in turn composite objects, and the composition hierarchy can the be extended recursively with the sub-hierarchies corresponding to the composition hierarchies of the components. The term "hierarchy" is used here in the sense of a directed graph without cycles: the composition hierarchy is usually a tree but is necessarily a graph as soon as a component is shared by two composite objects. This is the case when a component models a separation or a frontier between two composite objects.

There are two standard points of view for representing composite objects in a frame-based representation system. In weak composition, the parts of a composite object are described by the attributes of the object. A composite object is then defined by a frame whose attributes refer to the components. The attributes are used as composition links to access the components and each component must be explicitly placed in the attribute value. A special initialization method has to be written to create the required instances and to place the appropriate values. The weak composition approach is simple and can be simulated in every frame-based representation system. However, this simplicity has some drawbacks: the composition relation is distributed among the attributes and cannot be considered on its own and be controlled in consequence.

By contrast, in strong composition, the composition relation is explicitly represented and its properties as well. A composite object can then be defined by a frame with a special "composition" attribute recording the list of the components and the composition links used to access these components (for reading or writing values). Hence, the composition relation is particularized and can be considered on its own. Moreover, a composite object is handled as a "whole".

In both perspectives, a composite object shares properties with its components. Property sharing between the parts and the whole can be likened to inheritance. However, property sharing is not oriented as in the inheritance mechanism, but conflicts may appear and must be solved as inheritance conflicts are.

3 Composition in description logics

In the following, we briefly summarize the work presented in [Lambrix,1996] on the representation of the part-whole relation in description logics and the associated reasoning process.

Many interesting studies on composition in description logics can be found in [Speel and Patel-Schneider,1994] [Sattler,1995] [Artale *et al.*,1996] [Lambrix,1996] and [Rousset and Hors,1996]. The works presented in [Lambrix,1996] (and in [Padgham and Lambrix,1994]) is one of the most complete. It describes a model for representing the composition relation in description logics and for reasoning with composite descriptions. The author introduces also a number of operations to be performed on composite descriptions, mainly about instantiation of composite objects and recognition of specific parts. More precisely, compositional inference, compositional extension and completion are operations aimed at inferring whether an individual can be built with the help of other individuals.

These three operations are *composes*, *assembly* and *candidate completion*. The composes relation is used to check whether a set of individuals can be used as parts and modules for an individual belonging to a particular concept. The assembly operation extends a given knowledge base by inferring new individuals given other individuals that can be used as parts. Candidate completion is used to find out what parts are still missing to be able to instantiate an individual.

4 Structural descriptions

Structural descriptions have been studied and used in the field of pattern recognition, especially for matching and recognition purposes [Haralick and Shapiro,1993] [Vosselman,1992]. Structural descriptions can be seen as a set of primitive objects, the components or parts, interconnected through a set of relations to form a composite description, the whole. Structural descriptions can be a basis for representing complex composite real-world concepts as structures. Structural descriptions are used to guide a recognition process. The knowledge base of a recognition system includes structural descriptions of prototype objects, used as models during the recognition process. The system analyzes candidate descriptions by computing their structural descriptions and by trying to match every candidate with a model.

A structural description of a real-world concept $\mathtt{C}$ is a pair $\mathtt{D} = \{\mathtt{P}, \mathtt{R}\}$, where $\mathtt{P}$ represents the parts of $\mathtt{C}$ and $\mathtt{R}$ the interrelationships among those parts. Intuitively, the set $\mathtt{P}$ represent the parts of a concept and the set $\mathtt{R}$ represents the interrelationships among the parts. Structural descriptions can be viewed as composite objects: parts are the primitive elements in the set $\mathtt{P}$ and interrelationships between parts are the named relations in the set $\mathtt{R}$.

5 Discussion: structures, composite objects and structural descriptions

A composite object can be described by a structural description. However, to be able to represent any kind of physical composition, it is necessary to extend the model of structural descriptions in the following way. The elements of $\mathtt{P}$ must be primitive components that cannot be decomposed ; thus we define a *generalized structural descriptions* as a structural description where elements in the set $\mathtt{P}$ can be structural descriptions themselves.

The composition information is given by the sets of parts $\mathtt{P}$ and the sets of named relations $\mathtt{R}$. The set $\mathtt{R}$ is divided into the set of all direct composition relations and the set of configuration relations. Thus, the *structural composition relation* is given by the set union $\mathtt{composition} + \mathtt{configuration}$; the structural composition relation corresponds to the transitive closure of the "direct composition" relation. Moreover, a component has a name corresponding to the root of the composition hierarchy attached to this component and that several components of the same type may exist.

Considering two composite objects $\mathtt{A}$ and $\mathtt{B}$ and their structural descriptions denoted by $(\mathtt{P}_\mathtt{A}, \mathtt{R}_\mathtt{A})$ and $(\mathtt{P}_\mathtt{B}, \mathtt{R}_\mathtt{B})$, the structural composition relation is aimed at comparing these structural descriptions, verifying either $\mathtt{A}$ is a "substructure" of $\mathtt{B}$, or the converse. Let $\mathtt{A}$ and $\mathtt{B}$ be two composite objects, $(\mathtt{P}_\mathtt{A}, \mathtt{R}_\mathtt{A})$ and $(\mathtt{P}_\mathtt{B}, \mathtt{R}_\mathtt{B})$ their structural descriptions: $\mathtt{A}$ is a *structural component* of $\mathtt{B}$ if and only if there exists an exact matching from $\mathtt{A}$ to $\mathtt{B}$. The relation between $\mathtt{A}$ and $\mathtt{B}$ is denoted by $\mathtt{B} \preceq \mathtt{A}$, and $\mathtt{A}$ is said to *subsume* $\mathtt{B}$ *with respect to the structural composition relation.*

The structural composition relation is a partial ordering, and it can be used to organize structural descriptions in a hierarchy $\mathcal{P}$ (*partonomy*). Since the structural composition relation is a partial ordering, the classification process can be applied on the basis of this relation to draw inferences and to solve problems involving composite objects.

Graphs are a special kind of structures and are a special kind of structural descriptions. Thus, we can define a class $\mathtt{class(s)}$ as the class of graphs including the graph $\mathtt{s}$, i.e. $\mathtt{s}$ is a structural component of every graph $\mathtt{g} \in \mathtt{class(s)}$. The relation $\preceq$ is used to organize graphs in a hierarchy $\mathcal{G}$ that is supposed to be constituted of a set of predefined graphs defining the reference classes of $\mathcal{G}$ and of a root Λ such that $\forall \mathtt{g} \in \mathcal{G}, \mathtt{g} \preceq \Lambda$. More precisely, given an individual graph $\mathtt{i}$, the class of $\mathtt{i}$ denoted by $\mathtt{class(i)}$ is defined by the most specific class $\mathtt{class(s)}$

in $\mathcal{G}$ such that s is a structural component of i: there does not exist s' such that $i \preceq s' \wedge class(s') \preceq class(s)$ (dually, if $i \in class(s)$, then s is the "largest" component of $\mathcal{G}$ included in i). For the sake of simplicity, we suppose that the class class(i) for an individual i is not a disconnected graph, and that class(i) is unique for every individual i.

The classification cycle can be used in $\mathcal{G}$ for three main reasoning purposes: checking a composition relation, property sharing and component type recognition. The *composition relation checking* operation consists in checking whether a composition relation may exist between two given individuals i and j. This will be the case if $class(i) \preceq class(j)$ or the converse is true. The *property sharing* rule between graphs in $\mathcal{G}$ follows from the hierarchical organization of $\mathcal{G}$. An individual i inherits a property prop, e.g. an attribute-value pair, if and only if $class(i) \preceq class(j)$ and prop is attached to class(j). The *component type checking* operation consists in checking the types of potential composite graphs a given individual i may be a component, and the types of potential components of i. This operation is based on the classification process: i may be a component of j as soon as $class(j) \preceq class(i)$ in $\mathcal{G}$; i may have a component j as soon as $class(i) \preceq class(j)$ in $\mathcal{G}$.

Composition relation checking and component type checking can be likened to the reasoning operations detailed in [Lambrix,1996]. Other operations applying to structures are also described from a knowledge acquisition point of view in [Breuker and de Velde,1994]. A precise comparison has still to be carried out.

References

[Artale *et al.*, 1996] A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-whole relations in object-oriented systems: An overview. *Data & Knowledge Engineering*, 20:347–383, 1996.

[Breuker and de Velde, 1994] J. Breuker and W. Van de Velde, editors. *ComonKADS Library for Expertise Modelling*. IOS Press, Amsterdam, 1994.

[Haralick and Shapiro, 1993] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision (Volume I and II)*. Addison Wesley, Reading, Massachusetts, 1993.

[Lambrix, 1996] P. Lambrix. *Part-Whole Reasoning in Description Logics*. PhD thesis, Deparment of Computer and Information Science, University of Linköping, Sweden, 1996.

[Napoli *et al.*, 1994] A. Napoli, C. Laurenço, and R. Ducournau. An object-based representation system for organic synthesis planning. *International Journal of Human–Computer Studies*, 41(1/2):5–32, 1994.

[Padgham and Lambrix, 1994] L. Padgham and P. Lambrix. A Framework for Part-of Hierarchies in Terminological Logics. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94), Bonn, Germany*, pages 485–496, 1994.

[Rousset and Hors, 1996] M.-C. Rousset and P. Hors. Modeling and Verifying Complex Objects: A Declarative Approach Based on Description Logics. In *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI'96), Budapest, Hungary*, pages 328–332, 1996.

[Sattler, 1995] U. Sattler. A Concept Language for an engineering application with part-whole relations. In A. Borgida, M. Lenzerini, D. Nardi, and B. Nebel, editors, *Proceedings of the 1995 International Workshop on Description Logics, Universita di Roma, June 2–3 (Technical Report 07.95)*, pages 119–123, 1995.

[Speel and Patel-Schneider, 1994] P.-H. Speel and P.F. Patel-Schneider. A Whole-Part Extension for Description Logics. In *Proceedings of the* ECAI'94 *Workshop on Parts and Wholes (Conceptual Part-Whole Relations and Formal Mereology), Amsterdam*, pages 111–121, 1994.

[Vosselman, 1992] G. Vosselman. *Relational matching*. Lecture Notes in Computer Science 628. Springer-Verlag, Berlin, 1992.