A Proposal for a Layered Architecture for a Hybrid Object-Based Representation System

Amedeo Napoli
CRIN CNRS–INRIA Lorraine
B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France
(Email: napoli@loria.fr)

Position paper for the 1996 International Workshop on Description Logics, Boston, November 1996.

1 Introduction

In this position paper, we present a proposal for a layered architecture for a *hybrid object-based representation system*, relying on two kinds of representation formalisms, namely description logics and frame knowledge representation systems or frame-based systems[1].

In description logics, concepts, roles and individuals represent real-world concepts, their properties and their instances. The subsumption relation is used to organize concepts and roles in hierarchies. The subsumption relation is the basis of the two main reasoning mechanisms, classification and instantiation. By contrast, in a frame-based system, real-world knowledge is represented by frames –or classes– and instances. A frame has an identity, a state and a behavior, and it is used to generate a set of instances. Frames are organized in a hierarchy relying on the specialization relation, where the inheritance mechanism is used for information retrieval and for default reasoning. Moreover, the behavior of frames, i.e. methods, can be activated by message passing.

The layered architecture described in the following is a first practical framework for integrating multiple representation and programming formalisms, namely constructions and knowledge units of frame-based systems and description logics. This architecture has been inspired by knowledge representation systems such as TROPES, aimed at representing viewpoints [Euzenat,1993], the reflective systems 3-KRS [Maes,1987b] [Maes,1987a] and KRS [Van Marcke,1988], the distributed system CodA [McAffer,1995], the CLOS language [Keene,1989] (mainly the fact that generic functions are no longer attached to classes), and systems based on *multi-instantiation* [Bertino and Guerrini,1995]. This work is also the continuation of studies on the integration of multiple representation and programming formalisms presented during two previous Description Logics Workshops [Napoli,1994] [Napoli,1995].

This integration tries to overcome limitations of description logics regarding representation, e.g. role value maps and behaviors, and limitations of frame-based systems, e.g. semantics and reasoning. The integration relies on *layers* corresponding to "degrees of complexity" regarding the knowledge representation purpose. Actually, only four layers are considered[2]. The first layer is related with a basic concept language, e.g. FL- [Levesque and Brachman,1987] or CLASSIC [Brachman *et al.*,1991] (the expression "concept language" refers to the language used to describe real-world concepts and relations). The second layer corresponds to a general concept language, e.g. BACK [Hoppe *et al.*,1993], KRIS [Baader and Hollunder,1991] or LOOM [MacGregor,1988]. The third layer associates functions free of side effects with the concepts of the second layer. The fourth layer corresponds to a general frame-based language, where concepts are

[1] The expression is borrowed from [Karp,1993]. Other expressions that could be used are object-oriented systems or object-based representation systems.

[2] This is a practical limitation. One could think of a greater number of layers: in the CodA system, seven meta-objects are attached to a single object; in reflective systems, the "tower" of meta-concepts can be composed of an indefinite number of components.

considered as frames defined by a state and a behavior, and where message passing and access-oriented programming are allowed, together with reasoning mechanisms such as classification.

The plan of this position paper is as follows. First, we introduce the TROPES system on which is based the present layered architecture. Then, we give details on the four layers, on the manipulation of knowledge through classification and message passing. Lastly, we briefly discuss the benefits of this layered approach to knowledge representation, before concluding the paper.

2 A Layered Architecture

2.1 Preliminaries: Viewpoints in TROPES

In this section, we emphasize the characteristics of TROPES that are used in the following. The TROPES system is a frame-based system designed to take into account multiple viewpoint representations [Mariño *et al.*,1990] [Euzenat,1993] [Tropes,1995]. A concept is represented by a set of related viewpoints, where a viewpoint is composed of a tree of classes (frames). Classes are themselves composed of a set of attributes that can be annotated by facets indicating the type or the domain of the attribute. No methods can be attached to a class but if-needed demons can be used to compute attribute values.

A class `C1` in the viewpoint `P1` can be related to a class `C2` in the viewpoint `P2` through an *oriented bridge* if and only if every instance of `C1` is an instance of `C2`, i.e. the oriented bridge materializes the fact that the extension of `C1` –the set of instances of `C1`– is included in the extension of `C2`. If two oriented bridges exist in the two inverse directions between the classes `C1` and `C2`, then `C1` and `C2` are equivalent, i.e. they have the same extension. One can note that one of the main principles of object-oriented programming, namely *mono-instantiation*, is no longer valid in TROPES.

Reasoning in TROPES is based on multiple viewpoint classification of instances, and proceeds in a classical way within a single viewpoint. When no more information is available in a viewpoint `P`, the classification process tries to take advantage of bridges whenever possible, going from the viewpoint `P` to a viewpoint `P'`, through a bridge, and trying to exploit information that was not usable in `P`. More precisely, whenever an instance `i` is classified in the extension of a class `C`, `i` becomes an instance of all classes that are related to `C` by a bridge. In this way, classification can be made more precise using every piece of information available.

2.2 From Viewpoints to Layers

Relying on the TROPES architecture, a layered architecture for a hybrid object-based representation system can be based on the the following principles:

- The hybrid system is organized around four layers, where a layer corresponds to a viewpoint on the complexity of the representation language used. Each layer is composed by a hierarchy of *(knowledge) units* representing real-world concepts, according to the complexity of the viewpoint.
- A real-world concept is represented by four related units belonging to each layer. The complexity and the type of a unit `U` depends on the layer in which lies `U`. Moreover, an individual – or an instance– is an instance of four units, and each related units have the same extension.
- Different characteristics regarding knowledge representation and reasoning are associated with each layer. Units are manipulated by classification and message passing. Instances are manipulated by an instantiation process or by message passing. Moreover, side effects are only handled in the fourth layer.

2.3 Details on the Layers

The First Layer: Basic Units

The first layer is related to a basic concept language, that can be FL- or CLASSIC. Primitive concepts, defined concepts, and roles are available. Actually, the set of available constructors for the concept language can be chosen according to the theoretical results given in [Donini *et al.*,1991b]. Thus, this first layer is characterized by a restricted concept language to which is associated a correct, complete and polynomial subsumption algorithm.

The manipulation of units and instances is based on a tell-ask interface and the reasoning services are based on classification and instantiation. The semantics of concepts is defined in a classical way and all operations performed on concepts, roles and individuals are in accordance with the semantics.

The Second Layer: Complex Units

The second layer is a natural extension of the first layer. The description of units can be completed with complex constructors such as disjunction or negation for defined concepts, role conjunction, etc. as this is the case for description logics such as BACK [Hoppe *et al.*,1993], KRIS [Baader and Hollunder,1991] or LOOM [MacGregor,1988]. The manipulation of concepts and instances is based on a tell-ask-forget interface, and the reasoning services are based on classification and instantiation, as this is already the case for the first layer. The theoretical results on complexity of subsumption apply to this layer [Donini *et al.*,1991a].

The Third Layer: A Meta-level

The third level can be likened to a meta-level. Units are composed of attributes and methods. Attributes record values that are valid for all instances of the unit, i.e. a kind of class variable. The set of methods define the behavior of the unit regarding representation and programming services, such as the meta-level components of CodA or the meta-concepts of KRS. The behavior includes methods for defining the instantiation of the unit, the way the unit send, receive and answer messages, methods for printing and displaying the unit, etc.

The manipulation of units and instances is based on message passing. However, the methods associated with the third level are free of side effects, and thus, the result of the execution of these methods is always in accordance with the semantics of the associated units lying in the first and second levels.

The Fourth Layer: Units with Behaviors

As in the third level, the manipulation of units and instances in the fourth level is based on message passing. The fourth layer allows the definition of methods activated by message passing and producing side effects, e.g. changing attribute values in units and in instances.

For the sake of simplicity, we will suppose that these side effects cannot change the classification of a unit. However, two cases can occur for instances. First, the execution of a method changes the attribute values without changing the reference units of the instance (the instantiation). Second, the execution of a method changes the reference units of the instance: the instance is then *updated* and is reclassified with the new attribute values. The update of an instance is based on the notion of *temporal objects* as presented in [Napoli,1992]: the value of an attribute is actually a list recording the whole *history* of the value modifications of the attributes. In this way, whenever necessary, e.g. in case of incoherence, it is always possible to forget side effects and to return to a past and correct situation.

2.4 The Manipulation of Knowledge

Reasoning

Reasoning is based on a multiple layer classification cycle inspired by the multiple viewpoint classification cycle in TROPES. The classification cycle is based on four steps:

(1) *Definition* of the entity `X` –a unit or an instance– to be classified according to the a starting layer `L` (the starting layer is by default the first layer). When `X` is a unit, three empty units are built in the three other layers, for making the classification coherent and parallel in the four layers.

(2) *Classification* of `X` in the layer `L`: classical search for the most specific subsumers (MSS) and the most general subsumees (MGS) of `X`; when `X` is an instance, the search for the MGS is useless.

(3) *Operations*: the insertion of `X` in the layer `L` activates update operations or message passing.

(4) *Transition*: if information about `X` is still available but is not usable in the current layer, the classification process continues in a more complex layer thanks to the bridges between units.

The classification process makes explicit the dependencies holding between `X` and the units lying in the hierarchy of a layer. A parallel can be drawn between the classification process for an instance `X` and the management of rules in description logics. When classification is performed in a given layer, knowledge attached to other layers is "ignored", but is attached to `X` when the classification is over, as this is the case for *incidental* knowledge associated with rules.

Programming

Programming is based on message passing as this is the case in the object-oriented paradigm. Units have behaviors that are defined in the third and fourth layers. Thus, it is possible to attach different

behaviors to different *type* of units, e.g. methods for initialization, instantiation, visualization, printing and message handling. In this way, it is possible to design a distributed behavior of units, depending on the type of units.

The fourth layer allows the definition of methods producing side effects. However, this complex subject must still be more deeply investigated in order to make this proposal correct and coherent from this point of view. Thus, we will not elaborate anymore on this subject.

3 Discussion and Conclusion

In this position paper, we have presented a practical proposal for a layered architecture for an hybrid object-based representation system, integrating description logics and frame systems features. The ideas introduced in this paper must still be made more precise and much work remains to be done, in order to design a well founded framework for the integration of both representation paradigms. Moreover, this proposal has still to be validated by an implementation (planned in a near future). This first proposal is imperfect and must be improved, but we believe that it is slightly different from other proposals and that it can be exploited with profit for integration purposes.

References

[Baader and Hollunder, 1991] F. Baader and B. Hollunder. KRIS: Knowledge Representation and Inference System. ACM SIGART BULLETIN, 2(3):8–14, 1991.

[Bertino and Guerrini, 1995] E. Bertino and G. Guerrini. Objects with Multiple Most Specific Classes. In *Proceedings of* ECOOP'95, *Aarhus, Denmark, Lecture Notes in Computer Sciences 952*, pages 102–126, Berlin, 1995. Springer-Verlag.

[Brachman *et al.*, 1991] R.J. Brachman, D.L. McGuinness, P.F. Patel-Schneider, L.A. Resnick, and A. Borgida. Living with CLASSIC: When and How to Use a KL-ONE Language. In J. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1991.

[Donini *et al.*, 1991a] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The Complexity of Concept Languages. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91), Cambridge*, pages 151–162, 1991.

[Donini *et al.*, 1991b] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable Concept Languages. In *Proceedings of the 12th* IJCAI, *Darling Harbour, Sydney, Australia*, pages 458–463, 1991.

[Euzenat, 1993] J. Euzenat. On a purely taxonomic and descriptive meaning for classes. In *Proceedings of the IJCAI'93 Workshop on Object-Based Representation Systems, A. Napoli editor*, pages 81–92, 1993. The proceedings are available as Rapport de Recherche CRIN 93-R-156, Nancy, France.

[Hoppe *et al.*, 1993] T. Hoppe, C. Kindermann, J.J. Quantz, A. Schmiedel, and M. Fischer. BACK V5 – Tutorial & Manual. KIT-REPORT 100, Technische Universitæt, Berlin, 1993.

[Karp, 1993] P.D. Karp. The Design Space of Frame Knowledge Representation Systems. Technical Note 520, SRI International, Menlo Park, 1993.

[Keene, 1989] S.E. Keene. *Object-Oriented Programming in Common Lisp. A Programmer's Guide to CLOS.* Addison Wesley, Reading, Massachusetts, 1989.

[Levesque and Brachman, 1987] H.J. Levesque and R.J. Brachman. Expressiveness and Tractability in Knowledge Representation and Reasoning. *Computational Intelligence*, 3(2):78–93, 1987.

[MacGregor, 1988] R. MacGregor. A Deductive Pattern Matcher. In *Proceedings of* AAAI *'88, St Paul, Minnesota*, pages 403–408, 1988.

[Maes, 1987a] P. Maes. Computational Reflection. Technical Report 87_2, Artificial Intelligence Laboratory, Vrije Universiteit, Brussel, 1987.

[Maes, 1987b] P. Maes. Issues in Computational Reflection. In P. Maes and D. Nardi, editors, *Meta-Level Architectures and Reflection*, pages 21–35. North-Holland, Amsterdam, 1987.

[Mariño *et al.*, 1990] O. Mariño, F. Rechenmann, and P. Uvietta. Multiple Perspectives and Clas-

sification Mechanism in Object-Oriented Representation. In *Proceedings of the 9th* ECAI, *Stockholm, Sweden*, pages 425–430, 1990.

[McAffer, 1995] J. McAffer. Meta Level Programming with CodA. In *Proceedings of* ECOOP'95, *Aarhus, Denmark, Lecture Notes in Computer Sciences 952*, pages 190–214. Springer-Verlag, Berlin, 1995.

[Napoli, 1992] A. Napoli. An Object-Based Approach to Computer-Aided Planning of Organic Syntheses. Rapport de Recherche 92-R-101, Centre de Recherche en Informatique de Nancy, 1992.

[Napoli, 1994] A. Napoli. Studies about the Integration of Classification-Based Reasoning and Object-Oriented Programming. In F. Baader, M. Lenzerini, W. Nutt, and P.F. Patel-Schneider, editors, *Working Notes of the 1994 Description Logics Workshop*, pages 60–62. DFKI Saarbruecken, 1994.

[Napoli, 1995] A. Napoli. Objects, Classes, Specialization and Subsumption. 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 52–55, 1995.

[Tropes, 1995] Projet Sherpa, Tropes 1.0: reference manual. Rapport interne, INRIA Rhône-Alpes, Grenoble, France, 1995.

[Van Marcke, 1988] K. Van Marcke. The Use and Implementation of the Representation Language KRS. Technical Report 88-2, Vrije Universiteit, Brussel, 1988.