The Notion of Inheritance in Object-Oriented Programming

Kenneth Baclawski and Bipin Indurkhya*

April 3, 2006

Recently, there has been a debate over what the notion of inheritance means [1, 9, 13, 15, 16]. For example, Winkler [16] points to the dichotomy between the concept-oriented view (COV) and the program-oriented view (POV) of classes and inheritance. Wegner [15] has referred to COV as "logical" and POV as "physical." According to COV, squares form a subclass of rectangles. But this leads to some major problems in POV, where the set of operations applicable to a subclass is seen as a monotonic extension of its superclass. This is because applying operations such as "set-width" to a square object would allow one to change the width of a square independently of its length. To remedy this situation, Winkler proposes an object-oriented view (OOV) of inheritance that replaces the term "class," "subclass" and "superclass" with "object type," "extension type" and "base type," respectively. Winkler makes it seem to be a dichotomy of two different, overlapping but not identical, views of inheritance that can be resolved by renaming the concepts involved. But it turns out to be a more complex issue on deeper reflection.

To realize the full complexity underlying the notion of inheritance, we must take a closer look at what Winkler has casually dubbed COV. Is inheritance from a conceptual point of view always (or often) unambiguous? Is a square always a rectangle? While one may be tempted to answer both these questions in the affirmative at first, a further reflection reveals that things are more subtle. For instance, if we were to just think of a class of all squares and a class of all rectangles, then obviously the former is a proper subclass of

*Northeastern University, College of Computer Science, Boston, Massachusetts 02115

the latter. However, a concept is more than a class of objects, it also includes the actions (operations) one can perform on the objects. In fact, according to one view of cognition and concepts, objects can only be defined by specifying the possible ways of acting on them [7, 10]. For instance, Piaget [10] showed that the child *constructs* the notion of object permanence in terms of his or her own actions.

In view of all this, the concept square, which is defined to be the class of all squares without any actions on them, is not the same as the concept square in which the objects are allowed to be shrunk or stretched. In fact, it has been found that children's concepts of square and rectangle undergo several transformations as the child's repertoire of operations increases [11, 12]. No one of these concepts of square has any special claim to being the *natural* one with the others being aberrations. In particular, the concept square that allows stretching and shrinking is not a subconcept of rectangle with the same operations because these operations act differently on the objects in the two classes. This is precisely the problem with POV that Winkler points out in his article. But the very same problem is also present in COV.

The difficulty in modeling the hierarchy of everyday concepts in a programming environment is best exemplified by the failure of attempts to find an epistemological foundation of the IS-A link of semantic networks [2, 3, 4]. In fact, in one of these articles [3], Brachman points out that there are six different kinds of generic-generic relations and four different kinds of generic-individual relations. Confusion and misunderstanding is unavoidable if all these different relations are grouped together under one label IS-A.

The point here is that concepts in the real world, which programs and databases attempt to model, do not come in neatly packaged hierarchies. This mistaken view is the result of restricting oneself to simple examples such as different kinds of birds [6] or persons and elephants [14]. Wegner, in fact, assumes that these "intuitive" hierarchies are mind-independent, preexisting, real-world concept hierarchies. We need to look only a little further to see how complex our everyday concept hierarchies can be [8].

When we look at the POV, we find that it essentially refers to a certain set of *mechanisms* provided by the designers of a language. With these mechanisms, it may be possible to model hierarchies of many real-world concepts in different ways using different *policies.* For example, Halbert and O'Brien [5] list quite a variety of uses that one can make of the inheritance mechanism. A subtype can be a specialization, a restriction, a realization of a partial supertype, one of a collection of subtypes that partition a supertype (taxonomy), an implementation technique for ensuring that the subtype obeys a certain protocol, or a means of adding auxiliary functions (in the case of multiple inheritance). However, they still make the assumption that there is a "natural" or "intuitive" concept hierarchy. Like Winkler, it is taken for granted that circle is a subtype of ellipse. A hierarchy in which ellipse is a subtype of circle is, therefore, considered an aberration ("nonstandard"). Nonetheless, the authors admit that these nonstandard uses of inheritance can be valuable.

We would like to affirm a stronger point of view in which there are no "standard" conceptual hierarchies. Given a domain and a specific purpose, certain concept hierarchies would be clearly preferable than others, but such policy decisions are best left to the users of the programming language. Of course, it may not be possible to model every possible concept hierarchy with a given set of inheritance mechanisms, and for that reason the designer of an object-oriented language may want to give some thoughts to which policies may or may not be implementable. But to try to make these mechanisms reflect a certain preexisting, operation-independent, "logical" hierarchy of concepts in the world would be similar to chasing a mirage.

This whole discussion can be summed up as follows. What programming language provides is a set of mechanisms. While these mechanisms certainly restrict what one can do in that language and what views of inheritance can be implemented there, they do not by themselves validate some view of inheritance or other. Classes, specializations, generalizations and inheritance are only concepts, and like other concepts they do not have a universal objective meaning but depend heavily on the objects involved and the kind of operations allowed on these objects. This implies that how inheritance is to be incorporated in a specific system is up to the designers of the system, and it constitutes a policy decision that must be implemented with the available mechanisms.

References

[1] P. Abrahams. Subject: Objectivism. *Comm. ACM*, 34(1):15–16, 1991. Letter to ACM Forum.

[2] R. Brachman. On the epistemological status of semantic networks. In N. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, New York, NY, 1979.

[3] R. Brachman. What IS-A is and isn't: An analysis of taxonomic link in semantic networks. *Computer*, 16(10):30–36, 1983.

[4] R. Brachman. "I lied about the trees" or, defaults and definitions in knowledge representation. *AI Magazine*, pages 80–93, Fall 1985.

[5] D. Halbert and P. O'Brien. Using types and inheritance in object-oriented programming. *IEEE Software*, 4(5):71–79, 1987.

[6] B. Henderson-Sellers and J. Edwards. The object-oriented systems life cycle. *Comm. ACM*, 33(9):142–159, 1990.

[7] B. Indurkhya. *Metaphor and Cognition.* Kluwer Academic, Dordrecht, Netherlands, 1992.

[8] G. Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. University of Chicago Press, Chicago, 1987.

[9] W. LaLonde and J. Pugh. Subclassing $\neq$ subtyping $\neq$ is-a. *J. of Object-Oriented Programming*, 3(5):57–62, 1991.

[10] J. Piaget. *The Construction of Reality in the Child.* Ballantine Books, New York, 1971.

[11] J. Piaget and B. Inhelder. *The Child's Conception of Space.* Norton, New York, 1967.

[12] J. Piaget, B. Inhelder, and A. Szeminska. *The Child's Conception of Geometry.* Norton, New York, NY, 1981.

[13] L. Tesler. Object-oriented approach. *Comm. ACM*, 34(8):13–14, August 1991.

[14] P. Wegner. The object-oriented classification paradigm. In B. Shriver and P. Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 479–560. MIT Press (Computer Systems Series), Cambridge, MA, 1987.

[15] P. Wegner. Concepts and paradigms of object-oriented programming. *OOPS Messenger*, 1(1):7–87, 1990.

[16] J. Winkler. Objectivism: “class” considered harmful. *Comm. ACM*, 35(8):128–130, August 1992.