Object Integrity & Sharing $\Longleftarrow$ True Object-Orientation

Harri Hakonen Ville Leppänen Timo Raita Tapio Salakoski
Jukka Teuhola

Department of Computer Science, University of Turku, and
Turku Centre for Computer Science (TUCS), Finland

Abstract

Sharing (*aliasing*) of (sub)objects is a natural concept in object oriented (OO) software design because of the reference semantics in OO languages. Sharing means that a subordinate object is referenced by several other objects. For consistency reasons, it is advantageous to have a single copy of the object which the others can refer to.

Supporting correctness of programs with language constructs is a contemporary trend in OO languages. Besides programming errors, unexpected behavior of an operation is typically due to an object, participating the operation, not being in a legal state. Such legal states are characterized by the *integrity of objects*. E.g. in the Eiffel language one can state class invariants to characterize all the legal states.

Sharing and at the same time guaranteeing integrity of objects can be problematic, if the integrity of those objects depends on the state of the shared subordinate object(s), since the referring objects may not be aware of each other. Changes made to the subordinate object must respect all invariants dealing with it at the time of change. I.e., we think that objects are simultaneously in multiple contexts, and each context imposes constraints on the object stated by the invariants of the contextual objects.

In the literature, this problem is tackled by trying to control where such aliases can appear (Hogg's Islands, Almeida's Balloon types), and more recently by attaching some kind of protection to references [1, 2, 3]. Instead of trying to limit the ability to share subordinate objects, we consider restricting the effects that can be caused via aliases by attaching (arbitrary) constraints to any reference. Our idea is to create such constrained views to a subordinate object that via those views or references one is not able to break invariants of referring objects.

References

[1] H. Hakonen, V. Leppänen, T. Raita, T. Salakoski, and J. Teuhola. Improving Object Integrity and Preventing Side Effects via Deeply Immutable References. In *Proceedings of sixth Fenno-Ugric Symposium on Software Technology, FUSST'99*, pages 139 – 150, 1999.

[2] G. Kniesel and D. Theisen. JAC – Java with transitive readonly access control, 1999. Intercontinental Workshop on Aliasing in Object-Oriented Systems, IWAOOS'99.

[3] J. Noble, J. Potter, D. Holmes, and J. Vitek. Flexible Alias Protection. In *Proceedings of ECOOP'98*, Brussels, Belgium, July 20 - 24 1998.