A Conclusion on Inheritance Anomaly

Andrew E. Santosa
Oracle Labs Australia

Abstract

In concurrent programming using an object-oriented program, programmers adding new methods in a subclass typically have to modify the code of the superclass, which inhibits reuse, a problem known as *inheritance anomaly*. There have been much efforts by researchers in the last two decades to solve the problem by deriving anomaly-free languages. Yet, these proposals have not ended up as practical solutions, thus one may ask why.

This talk presents the recently published work of Gramoli and Santosa [1], where the authors investigated the issue from two perspectives. From the theoretical perspective, the authors demonstrate that a freedom from inheritance anomaly necessitates a language where ensuring Liskov-Wing substitutability becomes a language containment problem, which in their formal modeling is PSPACE hard. This indicates that we cannot expect programmers to manually ensure that subtyping holds in an anomaly-free language. Anomaly freedom thus predictably leads to software bugs and the work throws doubt on the value of providing it. From the practical perspective, the problem is already solved. Inheritance anomaly is part of the general *fragile base class problem* of object-oriented programming, that arises due to code coupling in *implementation* inheritance. In modern software practice, the fragile base class problem is circumvented by interface abstraction to avoid implementation inheritance, and opting for composition as means for reuse.

References

[1] V. Gramoli and Andrew E. Santosa. Why inheritance anomaly is not worth solving. In *ICOOOLPS '14*. ACM, 2014.