**Concurrent Garbage Collection**  
Based on “The Garbage Collection Handbook”

By Richard Jones, Antony Hosking and Eliot Moss

Presented and summarized by Roman Kecher  
23/12/2014

The talk was based on chapter 15 of the book, which discusses the general concepts related to concurrent garbage collection, the new kind of problems that it introduces, and a number of different approaches to solving those problems.

First, a few different garbage collection variants are introduced: an incremental garbage collection, a mostly concurrent garbage collection, and finally the concurrent “on-the-fly” garbage collection. Those variants slowly introduce the audience to the new major issue associated with concurrent garbage collection – mostly referred to as “the lost object problem”; since the mutator operates concurrently with the collector, it is able to influence the connectivity graph and cause some of the objects to be missed by the collector.

The tricolor abstraction (which labels objects with white, grey and black colors) is described once more. It is denoted that objects may be lost only if two conditions hold simultaneously: (1) there is a reference from a black object to a white object, and (2) that white object is not reachable from any grey object. Therefore, it is sufficient to prevent at least one of these conditions from becoming true; the strong and weak invariants, contradicting conditions (1) and (2) respectively, are therefore introduced.

The last part of the chapter and talk discusses the different read/write barrier techniques that can be utilized in order to implement and maintain either the strong or weak invariants, under different assumptions and in various environments (such as a grey or black mutator, or an incremental update technique vs that of snapshot-at-the-beginning). Implementation details, such as card tables, are also discussed.

A curious (and original) case presented in the slides, is a famous story about Dijkstra et al., who have proposed a specific barrier technique for maintaining one of the aforementioned invariants. Their proposed solution relaxed the atomicity requirements of the barrier, and a behavioral correctness proof has also been provided. As it turned out, the implementation was actually flawed, and the lack of atomicity caused a specific race condition between the code of the barrier (run by the mutator) and that of the garbage collector, leading to a subtle bug. This case convinced Dijkstra and others that state-based (assertional) reasoning should be preferred for concurrent algorithms. This whole case is detailed fully in the presentation slides.

The discussions in the classroom revolved mostly about the differences between the various barrier techniques that have been introduced, and the numerous quality measures by which different concurrent garbage collection implementations should be compared.