Using Aliasing Information to Predict Real-Time Garbage Collection

Patrik Persson

Department of Computer Science, Lund University, Sweden
e-mail: Patrik.Persson@cs.lth.se

Recent advances in garbage collection scheduling allows languages with garbage collection, like Java, to be used for applications with hard real-time constraints [1]. To determine the required amount of garbage collection work, and thus ensure that processes can execute without running out of memory, a worst-case estimation of the amount of live (i.e., referenced) memory (at any point in time) is required. Other real-time garbage collectors use similar metrics [2].

We have designed an algorithm to estimate the maximum amount of live memory in a program [3]. For simple programs without recursively defined classes, a conservative (albeit pessimistic) estimate can be obtained by assuming all references to denote unique objects. To handle recursive classes, we require every such class to be annotated with a path bound, indicating the maximal number of instances that may be linked together in sequence. Such a path bound corresponds to the length of a linked list or the depth of a binary tree.

The algorithm is based on a classification of references, where references are classified as **links** (introducing cycles into class declarations, such as the 'next' reference in a list element), **entries** (other references to recursive classes, such as the 'first' reference in a list head), **redundant** (references which are always aliased by some other non-redundant reference), and **simple** (non-redundant references to non-recursive classes).

To prevent an overly pessimistic analysis, we require redundant references to be explicitly annotated as such. Such annotations distinguish e.g. a doubly linked list from a binary tree (they both have two recursive pointers, but one is redundant in the doubly linked list). Interesting related questions include:

- How should these annotations be expressed?
- How can we verify (statically and/or dynamically) that the annotations are correct (or at least consistent with the program)?

References

1. R. Henriksson. *Scheduling Garbage Collection in Embedded Systems.* Ph.D. thesis, Department of Computer Science, Lund University, Sweden, 1998.
2. K. Nilsen. Reliable Real-Time Garbage Collection of C++. *Computing Systems*, 1994.
3. P. Persson. Live Memory Analysis for Garbage Collection in Embedded Systems. In *Proceedings of the ACM SIGPLAN 1999 Workshop on Languages, Compilers, and Tools for Embedded Systems (LCTES'99)*, pages 45-54. Atlanta, GA, May 1999.