CXXR: Refactoring the R Interpreter into C++

Andrew R. Runnalls,
University of Kent, UK *

25 March 2008

CXXR (www.cs.kent.ac.uk/projects/cxxr) is a project to refactor (reengineer) the interpreter of the R language, currently written for the most part in C, into C++, whilst as far as possible retaining full functionality. It is hoped that by reorganising the code along object-oriented lines, by deploying the tighter code encapsulation that is possible in C++, and by improving the internal documentation, the project will make it easier for researchers to develop experimental versions of the R interpreter. The author's own medium-term objective is to create a variant of R with built-in facilities for provenance tracking, so that for any R data object it will be possible to determine exactly which original data files it was derived from, and exactly which sequence of operations was used to produce it. (In other words, an enhanced version of the old S AUDIT facility.)

At the time of this abstract:

- Memory allocation and garbage collection have now been decoupled from each other and from R-specific functionality, and encapsulated within C++ classes. Classes `CellPool`, `MemoryBank` and `Allocator` look after memory allocation; `GCManager`, `GCNode`, `GCRoot` and `WeakRef` look after garbage collection. (All CXXR classes are within the namespace `CXXR`.) Class `GCRoot` provides C++ programmers with a mechanism for protecting objects from the garbage collector, as a more user-friendly (and probably less error-prone) alternative to the PROTECT/UNPROTECT mechanism used in standard R
- The `SEXPREC` union of CR is being progressively converted into an extensible hierarchy of classes rooted at a class `RObject` (which inherits from `GCNode`). This has already happened for vector objects and CONS-cell type objects, and it is now straightforward to introduced new types of R object simply by inheriting from `RObject`.

The proposed paper will:

1. Describe the motivation behind CXXR;
2. Report on progress to date;
3. Illustrate some of the simplified coding practices that CXXR enables;
4. Describe the measures taken to keep CXXR in synch with successive releases of standard R;
5. Outline future plans.

The paper will assume some familiarity with C programming and with concepts of object-oriented programming (e.g. in R or in Java), but C++-specific concepts will be explained as required.

*Computing Laboratory, The University, Canterbury CT2 7NF. Email: `A.R.Runnalls@kent.ac.uk`