Table of Contents

This document provides a very high-level overview of some of the facilities avilable in the *RGCCTranslationUnit* package and some of the possible uses or applications of these tools. For a more example-oriented description of how to use some of the functions in this package, see intro.xml.

Overview

At its simplest, the package provides (via a Perl module and the RSPerl package) a mechanism for reading into R the contents of a TU (Translation Unit) file generated by gcc or g++. This means that we can manipulate the detailed descriptions of arbitrary C/C++ code from within R. The package tries to provide higher-level facilities to work with the individual nodes from the TU file. Rather, it provides functionality to find the routines, the class and other data type definitions, enumerations, non-local variables and so on. These functions include *getRoutines*() *getVariables*() *getClassNodes*() , *getClassMethods*() , *getDataStructures*() and *getEnumerations*() . All of these are intended to provide either the raw nodes of interest, or a simple description of the relevant native code construct, e.g. a routine. We build on these to turn them into R-level, complete descriptions.

In addition to these functions to find particular common types of nodes, there are also low-level generic functions for traversing the node list/graph. The main ones of these are *nodeIterator*() *getAll*() and *getNext*() . One can use these to search for different code characteristics, e.g. all assignments or calls to other routines.

R-level Descriptions

Given the node identifying a C++ class or a routine, we want more complete descriptions of the entire concept without having to walk to the sub-graph of nodes for that entity. The function *resolveType*() is used to resolve the different elements such as fields within a structure, parameters within a routine, etc. and create an R-level description which can be used independently of the TU nodes in Perl. (One can save these R objects and restore them in a subsequent session without having to use RSPerl and re-read the TU file.)

Additionally, it allows us to process the body of routines and methods