Building the Instances of Columbus Schema for C/C++ Preprocessing

László Vidács

Preprocessor directives are widely used in C/C++ programs and have various purposes. Virtually there is no real C program without file inclusion, macro expansion and conditional compilation. The preprocessor has proven useful to programmers for over two decades, but it has also a number of drawbacks. The fundamental problem about preprocessing from a program comprehension point of view is that the compiler gets the preprocessed code and not the original source code that the programmer writes. In many cases the two codes are quite different (according to a case study of UNIX software packages, 8.4% of the source code of the programs consist of preprocessor directives). Program code with lots of directives often causes difficulties in program understanding. To aid program comprehension we designed a C/C++ preprocessor schema which describes the usage of preprocessor directives in the source code. We also implemented a preprocessor which is able to generate schema instances from the source code. Using a schema instance the connection between the original source and the compiled source can be understood in concrete cases (for instance a macro expansion can be followed step-by-step from the macro call to the `#define` directive which defines it).

Conditional compilation allows the programmer to create several configurations in one source. Depending on the environment of the compilation the compiler gets different code, but always only one configuration (for example different code pieces belong to different operating systems). According to the conditional compilation we defined two kinds of schema instances: dynamic instances that describe directives inside one configuration, and static instances which are configuration-independent. Building dynamic instances is straightforward because the work of the preprocessor is followed accurately. However, a static instance shows relations also between configurations and can be built in various ways. Here the natural building strategy is the pessimistic approach (every possible relation is shown between directives), which can be much improved by dropping some unnecessary relations. Determining weather two directives belong to the same configuration is an important improvement to the building method of static instances. In our preprocessor, besides the pessimistic method, we experienced with more powerful building strategies as well.