1 Introduction

Extensible programming languages were first discussed in the 1960s. As originally discussed in Douglas McIlroy's 1960 paper, the original concept was to use a small number of macros in order to extend a compiler to accept very general extensions to the original language accepted by the compiler (CITE McIlroy's paper). This has been modernized in various ways, including the idea of extensible syntax, compilers, and even extensible runtimes (CITE? wiki has dead link).

One example of an extensible programming language is `ABLE`C (cite), an extensible version of the C programming language, built on top of the `Silver` (cite) attribute grammar system. Various extensions exist for `ABLE`C that were created to solve various problems (CITE paper containing extensions from MELT).

This paper examines the uses of extensible programming languages in a real-world application: an Internet Relay Chat (IRC) server implementation. The IRC protocol was originally designed for text-based conferencing (cite IRC RFC), and many server implementations are available. For this paper, we consider the specific implementation of `ngIRCd` (next-generation IRC daemon), written in the C programming language. There are several potential issues with this server implementation. First, any C program must include runtime checks to ensure that all pointers are not null. We introduce an `ABLE`C extension to allow for compile-time checking of possible null pointer dereferences. Another issue with the server is writing easily understandable I/O code in C. While the `epoll` API provided by the Linux kernel allows for some asynchronous I/O, we introduce another `ABLE`C extension to provide improved syntax for asynchronous I/O, akin to what Javascript provides. Finally, we introduce an extension to ensure that parsing code is secure. We do this utilizing the Wuffs programming language (CITE), which allows for compile-time checking for various parsing attacks which must be manually accounted for at run-time in C.