1 Non-Null Pointer Extension

1.1 Non-null Qualifiers

One of the most common sources of bugs in the C programming language is errors related to null pointers. The dereferencing of a null pointer is considered undefined behavior in C, and is undesirable when programming. In this section, we introduce an ABLEC extension to deal with null pointers (referred to herein as ABLEC-nonnull). This extension allows for compile-time checking for any possible null dereferences of pointers. This static analysis of a program is valuable, saving the programmer both time and effort when compared with the original C code.

1.2 Comparison

Consider the code here from the same function, one rewritten using ABLEC-nonnull. Note the difference in the two examples below.

```
/* return false on failure (realloc failure, invalid src/dest array) */
bool
array_copyb(array * dest, const char *src, size_t len)
{
        assert(dest != NULL);
        assert(src != NULL );

        if (!src || !dest)
                return false;

        array_trunc(dest);
        return array_catb(dest, src, len);
}
```

Figure 1: Function before applying the non-null extension

```
/* return false on failure (realloc failure, invalid src/dest array) */
bool
array_copyb(array * nonnull dest, const char * nonnull src, size_t len)
{

        array_trunc(dest);
        return array_catb(dest, src, len);
}
```

Figure 2: Function after applying the non-null extension

The first example is a simple function to copy a string into an array as it is written in the original

project. Note that if this function is passed a null pointer for either of the two pointer arguments, the `assert` statement will fail, causing a call to the `abort()` function, crashing our program.

Compare this to the function we get after applying the non-null extension. Here, we know that both pointer arguments are guaranteed to be non-null. Thus, we no longer need either of the assert statements, nor do we need the check in the `if` statement. This is a toy example implemented to illustrate the point that including this extension allows for shorter functions and fewer safety checks on the part of the end programmer.