Chapter 54: Constraints

Section 54.1: Duplicate variable names in the same scope

An example of a constraint as expressed in the C standard is having two variables of the same name declared in a scope1), for example:

```
void foo(int bar)
{
    int var;
    double var;
}
```

This code breaches the constraint and must produce a diagnostic message at compile time. This is very useful as compared to undefined behavior as the developer will be informed of the issue before the program is run, potentially doing anything.

Constraints thus tend to be errors which are easily detectable at compile time such as this, issues which result in undefined behavior but would be difficult or impossible to detect at compile time are thus not constraints.

1) exact wording:

```
Version = C99
```

If an identifier has no linkage, there shall be no more than one declaration of the identifier (in a declarator or type specifier) with the same scope and in the same name space, except for tags as specified in 6.7.2.3.

Section 54.2: Unary arithmetic operators

The unary `+` and `-` operators are only usable on arithmetic types, therefore if for example one tries to use them on a struct the program will produce a diagnostic eg:

```
struct foo
{
    bool bar;
};

void baz(void)
{
   struct foo testStruct;
   -testStruct; /* This breaks the constraint so must produce a diagnostic */
}
```