obviously implied by the normative text. Given the extra costs associated with C++03 exception specifications, there are no known implementations that took advantage of this freedom.

For C++11, the footnote was revised to refer only to permitting the use of the new **noexcept** exception specification, without further clarification of the normative text. There is, however, also general permission to add a nonthrowing exception specification to any nonvirtual C++ Standard Library function, and it might be inferred that this provision gives implementations freedom to add such specifications to their C library wrappers too. Also note that functions taking callbacks, such as `bsearch` and `qsort`, will propagate exceptions thrown by the callback.

Again, there are no known implementations taking advantage of this freedom to add a nonthrowing exception specification to C library functions, although all of the functions in the `<atomic>` header intended for C interoperability are declared as **noexcept**, exploiting an arguable interpretation of the intent of this footnote.

Constraints on the `noexcept` specification imposed for `virtual` functions

When using C++03-style dynamic exception specifications, the exception specification of any function override cannot be wider than that of the function being overridden:

```
struct BB03
{
    void n() throw();
    virtual void f();
    virtual void g1() throw();
    virtual void g2() throw();
    virtual void g3() throw();
    virtual void h()  throw(int, double);
};

struct DD03 : public BB03
{
    void n() throw(int);               // OK, hiding nonvirtual function
    void n(char) throw(int);           // OK, additional overload
    virtual void f();                  // OK, base lacks exception spec.
    virtual void g1() throw();         // OK, same exception spec
    virtual void g2() throw(int);      // Error, wider exception spec (int)
    virtual void g3();                 // Error, wider exception spec (all)
    virtual void h() throw(int);       // OK, tighter exception spec
};
```

Interestingly, the rules relating to **virtual** functions and **noexcept** are still defined by the C++11 and C++14 Standards in terms of dynamic exception specifications, despite that dynamic exception specifications are deprecated. The C++14 Standard states[9]:

[9] **iso11a**, section 15.4, "Exception specifications," paragraph 5, p. 406