Latest results from the extensions WG are that the "ideal" syntax for explicit qualification of member template functions is not implementable. For example, given:

```c++
struct A { template <class T> void f(); };
```

the syntax

```c++
A a; a.f<void>();
```

cannot be integrated into the language grammar. Rather, the following is required, under the status quo:

```c++
A a; a.template f<void>();
```

which while possible to use in odd corners of a library implementation, is not appropriate for the a public class interface.

Therefore, I propose to remove the locale member function templates `locale::use<>()` and `locale::has<>()` and replace them with function templates as follows:

```c++
namespace std {
  template <class Facet> Facet const& use_facet(locale const&);
  template <class Facet> bool has_facet(locale const&);
}
```

Given a locale named `loc` and a facet type named `Fac`, these are called as

```c++
use_facet<Fac>(loc).member(); // was: loc.template use<Fac>().member()
if (has_facet<Fac>(loc)) // was: if (loc.template has<Fac>())
```

I am not proposing changes to the constructor templates in `locale`.