Document number:	P2495R1
Date:	2022-09-14
Project:	Programming Language C++
Audience:	LEWG
Reply-to:	Michael Florian Hava[1] <mfh.cpp@gmail.com>

Interfacing stringstreams with string_view

Abstract

This paper proposes amending the interface of `basic_[i|o]stringstream` and `basic_stringbuf` to support construction and reinitialization from `basic_string_view`.

Tony Table

Before

```
const ios_base::openmode mode;
const allocator<char> alloc;
const string str;
//implicitly convertible to string_view
const mystring mstr;


stringstream s0{""};                                    ✔
stringstream s1{"", alloc};                             ✘
stringstream s2{"", mode, alloc};                       ✘

stringstream s3{""sv};                                  ✘
stringstream s4{""sv, alloc};                           ✘
stringstream s5{""sv, mode, alloc};                     ✘

stringstream s6{""s};                                   ✔
stringstream s7{""s, alloc};                            ✔
stringstream s8{""s, mode, alloc};                      ✔

stringstream s9{str};                                   ✔
stringstream s10{str, alloc};                           ✔
stringstream s11{str, mode, alloc};                     ✔

stringstream s12{mstr};                                 ✘
stringstream s13{mstr, alloc};                          ✘
stringstream s14{mstr, mode, alloc};                    ✘


stringstream s15;
s15.str("");                                            ✔
s15.str(""sv);                                          ✘
s15.str(""s);                                           ✔
s15.str(str);                                           ✔
s15.str(mstr);                                          ✘



//concerning LWG2946
stringstream s16({"abc", 1});                           ✔
stringstream s17({"abc", 1}, alloc);                    ✘
stringstream s18({"abc", 1}, mode, alloc);              ✘


stringstream s19;
s19.str({"abc", 1});                                    ✔
```

Proposed

```
const ios_base::openmode mode;
const allocator<char> alloc;
const string str;
//implicitly convertible to string_view
const mystring mstr;


stringstream s0{""};                                    ✔
stringstream s1{"", alloc};                             ✘
stringstream s2{"", mode, alloc};                       ✘

stringstream s3{""sv};                                  ✔
stringstream s4{""sv, alloc};                           ✔
stringstream s5{""sv, mode, alloc};                     ✔

stringstream s6{""s};                                   ✔
stringstream s7{""s, alloc};                            ✔
stringstream s8{""s, mode, alloc};                      ✔

stringstream s9{str};                                   ✔
stringstream s10{str, alloc};                           ✔
stringstream s11{str, mode, alloc};                     ✔

stringstream s12{mstr};                                 ✔
stringstream s13{mstr, alloc};                          ✔
stringstream s14{mstr, mode, alloc};                    ✔


stringstream s15;
s15.str("");                                            ✔
s15.str(""sv);                                          ✔
s15.str(""s);                                           ✔
s15.str(str);                                           ✔
s15.str(mstr);                                          ✔



//concerning LWG2946
stringstream s16({"abc", 1});                           ✔
stringstream s17({"abc", 1}, alloc);                    ✘
stringstream s18({"abc", 1}, mode, alloc);              ✘


stringstream s19;
s19.str({"abc", 1});                                    ✔
```


[1] RISC Software GmbH, Softwarepark 32a, 4232 Hagenberg, Austria, michael.hava@risc-software.at

Revisions

R0: Initial version

R1: Updates after LEWG Review on 2022-08-16:

- Evaluated LWG2946 based on LEWG feedback.
  - Adjusted proposed design & wording accordingly.
  - Removed evaluation of alternative designs as they are either incompatible with LWG2946 or result in an ABI-break.
  - Dropped support for construction from `const CharT *` with an allocator and an optional `openmode`.
- Drive-by fix in [istringstream.cons]: added missing Constraints.
- Added section with frequently asked questions.

Motivation

[string.view] specifies `basic_string_view`, a vocabulary type template that represents an immutable reference to some string-like object. Unless a string can be moved from source to target, it is generally advisable to pass "immutable stringy inputs" by `basic_string_view`. Doing so obviates the need for multiple overloads and enables support for user-defined types.

[string.streams] specifies the class templates `basic_[i|o]stringstream` and `basic_stringbuf` to represent streams operating on/buffers owning a string. These classes predate the introduction of `basic_string_view` and therefore only support `basic_string` in their interfaces. Partial support for raw strings is provided by implicitly constructing a `basic_string` and then moving it.

This leads to an embarrassing problem when following the aforementioned recommendation: Every `basic_string_view` and user-defined string type must be explicitly converted to a temporary `basic_string` that is then moved into the respective constructor/member function. This paper aims to solve these issues by introducing direct support for `basic_string_view`.

Design space

As all classes in [string.streams] adhere to the following fragment for the context of construction/reinitialization from a string, the potential design is presented in terms of CLASS:

```
template<typename CharT, typename Traits, typename Alloc>
struct CLASS {
  //constructors interfacing with stringy inputs
  explicit CLASS(const basic_string<CharT, Traits, Alloc>&, ios_base::openmode = /*def*/);                1

  template<typename SAlloc>
  CLASS(const basic_string<CharT, Traits, SAlloc>&, const Alloc&);                                        2

  template<typename SAlloc>
  CLASS(const basic_string<CharT, Traits, SAlloc>&, ios_base::openmode, const Alloc&);                    3

  template<typename SAlloc>
  requires(!std::is_same_v<Alloc, SAlloc>)
  explicit CLASS(const basic_string<CharT, Traits, SAlloc>&, ios_base::openmode = /*def*/);               4

  explicit CLASS(basic_string<CharT, Traits, Alloc>&&, ios_base::openmode = /*def*/);                     5


  //reinitialization of internal string
  void str(const basic_string<CharT, Traits, Alloc>&);                                                    6

  template<typename SAlloc>
  requires(!std::is_same_v<Alloc, SAlloc>)
  void str(const basic_string<CharT, Traits, SAlloc>&);                                                   7

  void str(basic_string<CharT, Traits, Alloc>&&);                                                         8
```

The constructor and member function overloads can roughly be classified as follows:

No	Description
1	Copying the string.
2 3	Copying the string, input may have different allocator. Invalid for `const CharT *`.
4	Equal to 1 but input has different allocator. Invalid for `const CharT *`.
5	Moving the string, used for `const CharT *`.
6	Copying the string.
7	Equal to 6 but input has different allocator. Invalid for `const CharT *`.
8	Moving the string, used for `const CharT *`.

I propose to add restricted `basic_string_view`-overloads for 1 2 3 6:

```
template<typename T>
static
constexpr
bool is_string_view_like_v{std::is_convertible_v<const T&, std::basic_string_view<CharT, Traits>> &&
                           !std::is_convertible_v<const T&, const CharT*>}; //exposition only


//add to existing class definition:
template<typename T>
requires is_string_view_like_v<T>
explicit CLASS(const T&, ios_base::openmode = /*def*/);

template<typename T>
requires is_string_view_like_v<T>
CLASS(const T&, const Alloc&);

template<typename T>
requires is_string_view_like_v<T>
CLASS(const T&, ios_base::openmode, const Alloc&);

template<typename T>
requires is_string_view_like_v<T>
void str(const T&);
```

Due to following the design of LWG2946, constructions with `const CharT *`, an allocator, and an optional `openmode` (akin to 2 3) remains unsupported.

Impact on the Standard

This proposal is a pure library addition. Existing standard library classes are modified in a non-ABI-breaking way. Overload resolution for existing code is not affected by the introduced overloads.

Implementation Experience

The proposed overload set has been implemented on [https://godbolt.org/z/vo5c5P6eT] for evaluation[2]. Additionally, the proposed design has been implemented on a fork of the MS-STL [https://github.com/MFHava/STL/tree/P2495].

[2] An updated evaluation of all overload sets presented in R0 can be found here: https://godbolt.org/z/esWWr6hTr

Frequently Asked Questions

Why is this needed when C++23 includes spanstream?

Whilst there certainly is an overlap between `basic_spanstream` and `basic_stringstream`, fundamental differences in their semantics (ownership & growability) preclude the former to be a drop-in replacement for all conceivable uses of the latter.

Proposed Wording

Wording is relative to [N4910]. Additions are presented like this, removals like ~~this~~.

[version.syn]

In [version.syn], add:

```
#define __cpp_lib_sstream_from_string_view YYYYMML //also in <sstream>
```

Adjust the placeholder value as needed to denote this proposal's date of adoption.

[stringbuf.general]

In [stringbuf.general], in the synopsis, add the proposed overloads:

```
...
// 31.8.2.2, constructors
basic_stringbuf() : basic_stringbuf(ios_base::in | ios_base::out) {}
explicit basic_stringbuf(ios_base::openmode which);
explicit basic_stringbuf(
  const basic_string<charT, traits, Allocator>& s,
  ios_base::openmode which = ios_base::in | ios_base::out);
explicit basic_stringbuf(const Allocator& a)
  : basic_stringbuf(ios_base::in | ios_base::out, a) {}
basic_stringbuf(ios_base::openmode which, const Allocator& a);
explicit basic_stringbuf(
  basic_string<charT, traits, Allocator>&& s,
  ios_base::openmode which = ios_base::in | ios_base::out);
template<class SAlloc>
  basic_stringbuf(
    const basic_string<charT, traits, SAlloc>& s, const Allocator& a)
    : basic_stringbuf(s, ios_base::in | ios_base::out, a) {}
template<class SAlloc>
  basic_stringbuf(
    const basic_string<charT, traits, SAlloc>& s,
    ios_base::openmode which, const Allocator& a);
template<class SAlloc>
  explicit basic_stringbuf(
    const basic_string<charT, traits, SAlloc>& s,
    ios_base::openmode which = ios_base::in | ios_base::out);
template<typename T>
  explicit basic_stringbuf(const T& t, ios_base::openmode which = ios_base::in | ios_base::out);
template<typename T>
  basic_stringbuf(const T& t, const Allocator& a);
template<typename T>
  basic_stringbuf(const T& t, ios_base::openmode which, const Allocator& a);
basic_stringbuf(const basic_stringbuf&) = delete;
basic_stringbuf(basic_stringbuf&& rhs);
basic_stringbuf(basic_stringbuf&& rhs, const Allocator& a);

...
// 31.8.2.4, getters and setters
allocator_type get_allocator() const noexcept;

basic_string<charT, traits, Allocator> str() const &;
template<class SAlloc>
  basic_string<charT,traits,SAlloc> str(const SAlloc& sa) const;
basic_string<charT, traits, Allocator> str() &&;
basic_string_view<charT, traits> view() const noexcept;

void str(const basic_string<charT, traits, Allocator>& s);
template<class SAlloc>
  void str(const basic_string<charT, traits, SAlloc>& s);
void str(basic_string<charT, traits, Allocator>&& s);
template<typename T>
  void str(const T& t);
```

[stringbuf.cons]

In [stringbuf.cons]:

```
        template<class SAlloc>
          explicit basic_stringbuf(
            const basic_string<charT, traits, SAlloc>& s,
            ios_base::openmode which = ios_base::in | ios_base::out);
    8       Constraints: is_same_v<SAlloc, Allocator> is false.
    9       Effects: Initializes the base class with basic_streambuf() (31.6.3.2), mode with which, and buf with s, then calls init_buf_ptrs().

        template<typename T>
          explicit basic_stringbuf(const T& t, ios_base::openmode which = ios_base::in | ios_base::out);
   10       Constraints:
(10.1)      — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(10.2)      — is_convertible_v<const T&, const charT*> is false.
   11       Effects: Initializes the base class with basic_streambuf() (31.6.3.2), mode with which, and buf with t, then calls init_buf_ptrs().

        template<typename T>
          basic_stringbuf(const T& t, const Allocator& a);
   12       Constraints:
(12.1)      — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(12.2)      — is_convertible_v<const T&, const charT*> is false.
   13       Effects: Equivalent to basic_stringbuf(t, ios_base::in | ios_base::out, a).

        template<typename T>
          basic_stringbuf(const T& t, ios_base::openmode which, const Allocator& a);
   14       Constraints:
(14.1)      — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(14.2)      — is_convertible_v<const T&, const charT*> is false.
   15       Effects: Initializes the base class with basic_streambuf() (31.6.3.2), mode with which, and buf with {t,a}, then calls init_buf_ptrs().

        basic_stringbuf(basic_stringbuf&& rhs);
```

[stringbuf.members]

In [stringbuf.members]:

```
        void str(basic_string<charT, traits, Allocator>&& s);
   17       Effects: Equivalent to:
              buf = std::move(s);
              init_buf_ptrs();

        template<typename T>
          void str(const T& t);
   18       Constraints:
(18.1)      — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(18.2)      — is_convertible_v<const T&, const charT*> is false.
   19       Effects: Equivalent to:
              buf = t;
              init_buf_ptrs();
```

[istringstream.general]

In [istringstream.general], in the synopsis, add the proposed overloads:

```
...
// 31.8.3.2, constructors
basic_istringstream() : basic_istringstream(ios_base::in) {}
explicit basic_istringstream(ios_base::openmode which);
explicit basic_istringstream(
  const basic_string<charT, traits, Allocator>& s,
  ios_base::openmode which = ios_base::in);
basic_istringstream(ios_base::openmode which, const Allocator& a);
explicit basic_istringstream(
  basic_string<charT, traits, Allocator>&& s,
  ios_base::openmode which = ios_base::in);
template<class SAlloc>
  basic_istringstream(
    const basic_string<charT, traits, SAlloc>& s, const Allocator& a)
    : basic_istringstream(s, ios_base::in, a) {}
template<class SAlloc>
  basic_istringstream(
    const basic_string<charT, traits, SAlloc>& s,
    ios_base::openmode which, const Allocator& a);
template<class SAlloc>
  explicit basic_istringstream(
    const basic_string<charT, traits, SAlloc>& s,
    ios_base::openmode which = ios_base::in);
template<typename T>
```

```
  explicit basic_istringstream(const T& t, ios_base::openmode which = ios_base::in);
template<typename T>
  basic_istringstream(const T& t, const Allocator& a);
template<typename T>
  basic_istringstream(const T& t, ios_base::openmode which, const Allocator& a);
basic_istringstream(const basic_istringstream&) = delete;
basic_istringstream(basic_istringstream&& rhs);

...
// 31.8.3.4, members
basic_stringbuf<charT, traits, Allocator>* rdbuf() const;

basic_string<charT, traits, Allocator> str() const &;
template<class SAlloc>
  basic_string<charT,traits,SAlloc> str(const SAlloc& sa) const;
basic_string<charT, traits, Allocator> str() &&;
basic_string_view<charT, traits> view() const noexcept;

void str(const basic_string<charT, traits, Allocator>& s);
template<class SAlloc>
  void str(const basic_string<charT, traits, SAlloc>& s);
void str(basic_string<charT, traits, Allocator>&& s);
template<typename T>
  void str(const T& t);
```

[istringstream.cons]

In [istringstream.cons], additionally adding missing constraint:

```
        template<class SAlloc>
          explicit basic_istringstream(
            const basic_string<charT, traits, SAlloc>& s,
            ios_base::openmode which = ios_base::in);
      6     Constraints: is_same_v<SAlloc, Allocator> is false.
     6/7    Effects: Initializes the base class with basic_istream<charT, traits>(addressof(sb)) (31.7.4.2), and sb with basic_string-
            buf<charT, traits, Allocator>(s, which | ios_base::in) (31.8.2.2).

        template<typename T>
          explicit basic_istringstream(const T& t, ios_base::openmode which = ios_base::in);
      8     Constraints:
    (8.1)   — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
    (8.2)   — is_convertible_v<const T&, const charT*> is false.
      9     Effects: Initializes the base class with basic_istream<charT, traits>(addressof(sb)) (31.7.4.2) and sb with basic_string-
              buf<charT, traits, Allocator>(t, which | ios_base::in) (31.8.2.2).

        template<typename T>
          basic_istringstream(const T& t, const Allocator& a);
     10     Constraints:
   (10.1)   — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
   (10.2)   — is_convertible_v<const T&, const charT*> is false.
     11     Effects: Equivalent to basic_istringstream(t, ios_base::in, a).

        template<typename T>
          basic_istringstream(const T& t, ios_base::openmode which, const Allocator& a);
     12     Constraints:
   (12.1)   — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
   (12.2)   — is_convertible_v<const T&, const charT*> is false.
     13     Effects: Initializes the base class with basic_istream<charT, traits>(addressof(sb)) (31.7.4.2) and sb with basic_string-
              buf<charT, traits, Allocator>(t, which | ios_base::in, a) (31.8.2.2).

        basic_istringstream(basic_istringstream&& rhs);
```

[istringstream.members]

In [istringstream.members]:

```
        void str(basic_string<charT, traits, Allocator>&& s);
      8     Effects: Equivalent to: rdbuf()->str(std::move(s));

        template<typename T>
          void str(const T& t);
      9     Constraints:
    (9.1)   — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
    (9.2)   — is_convertible_v<const T&, const charT*> is false.
     10     Effects: Equivalent to: rdbuf()->str(t);
```

[ostringstream.general]

In [ostringstream.general], in the synopsis, add the proposed overloads:

```
...
// 31.8.4.2, constructors
basic_ostringstream() : basic_ostringstream(ios_base::out) {}
explicit basic_ostringstream(ios_base::openmode which);
explicit basic_ostringstream(
  const basic_string<charT, traits, Allocator>& s,
  ios_base::openmode which = ios_base::out);
basic_ostringstream(ios_base::openmode which, const Allocator& a);
explicit basic_ostringstream(
  basic_string<charT, traits, Allocator>&& s,
  ios_base::openmode which = ios_base::out);
template<class SAlloc>
  basic_ostringstream(
    const basic_string<charT, traits, SAlloc>& s, const Allocator& a)
    : basic_ostringstream(s, ios_base::out, a) {}
template<class SAlloc>
  basic_ostringstream(
    const basic_string<charT, traits, SAlloc>& s,
    ios_base::openmode which, const Allocator& a);
template<class SAlloc>
  explicit basic_ostringstream(
    const basic_string<charT, traits, SAlloc>& s,
    ios_base::openmode which = ios_base::out);
template<typename T>
  explicit basic_ostringstream(const T& t, ios_base::openmode which = ios_base::out);
template<typename T>
  basic_ostringstream(const T& t, const Allocator& a);
template<typename T>
  basic_ostringstream(const T& t, ios_base::openmode which, const Allocator& a);
basic_ostringstream(const basic_ostringstream&) = delete;
basic_ostringstream(basic_ostringstream&& rhs);

...
// 31.8.4.4, members
basic_stringbuf<charT, traits, Allocator>* rdbuf() const;

basic_string<charT, traits, Allocator> str() const &;
template<class SAlloc>
  basic_string<charT,traits,SAlloc> str(const SAlloc& sa) const;
basic_string<charT, traits, Allocator> str() &&;
basic_string_view<charT, traits> view() const noexcept;

void str(const basic_string<charT, traits, Allocator>& s);
template<class SAlloc>
  void str(const basic_string<charT, traits, SAlloc>& s);
void str(basic_string<charT, traits, Allocator>&& s);
template<typename T>
  void str(const T& t);
```

[ostringstream.cons]

In [ostringstream.cons]:

```
      template<class SAlloc>
        explicit basic_ostringstream(
          const basic_string<charT, traits, SAlloc>& s,
          ios_base::openmode which = ios_base::out);
   6    Constraints: is_same_v<SAlloc, Allocator> is false.
   7    Effects: Initializes the base class with basic_ostream<charT, traits>(addressof(sb)) (31.7.5.2), and sb with basic_string-
        buf<charT, traits, Allocator>(s, which | ios_base::out) (31.8.2.2).

      template<typename T>
        explicit basic_ostringstream(const T& t, ios_base::openmode which = ios_base::out);
   8    Constraints:
 (8.1)  — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
 (8.2)  — is_convertible_v<const T&, const charT*> is false.
   9    Effects: Initializes the base class with basic_ostream<charT, traits>(addressof(sb)) (31.7.5.2) and sb with basic_string-
          buf<charT, traits, Allocator>(t, which | ios_base::out) (31.8.2.2).

      template<typename T>
        basic_ostringstream(const T& t, const Allocator& a);
  10    Constraints:
(10.1)  — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(10.2)  — is_convertible_v<const T&, const charT*> is false.
  11    Effects: Equivalent to basic_ostringstream(t, ios_base::out, a).

      template<typename T>
        basic_ostringstream(const T& t, ios_base::openmode which, const Allocator& a);
  12    Constraints:
```

```
(12.1)      — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(12.2)      — is_convertible_v<const T&, const charT*> is false.
  13        Effects: Initializes the base class with basic_ostream<charT, traits>(addressof(sb)) (31.7.5.2) and sb with basic_string-
              buf<charT, traits, Allocator>(t, which | ios_base::out, a) (31.8.2.2).

      basic_ostringstream(basic_ostringstream&& rhs);
```

[ostringstream.members]

In [ostringstream.members]:

```
      void str(basic_string<charT, traits, Allocator>&& s);
   8      Effects: Equivalent to: rdbuf()->str(std::move(s));

      template<typename T>
        void str(const T& t);
   9      Constraints:
(9.1)       — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(9.2)       — is_convertible_v<const T&, const charT*> is false.
  10      Effects: Equivalent to: rdbuf()->str(t);
```

[stringstream.general]

In [stringstream.general], in the synopsis, add the proposed overloads:

```
...
// 31.8.5.2, constructors
basic_stringstream() : basic_stringstream(ios_base::out | ios_base::in) {}
explicit basic_stringstream(ios_base::openmode which);
explicit basic_stringstream(
  const basic_string<charT, traits, Allocator>& s,
  ios_base::openmode which = ios_base::out | ios_base::in);
basic_stringstream(ios_base::openmode which, const Allocator& a);
explicit basic_stringstream(
  basic_string<charT, traits, Allocator>&& s,
  ios_base::openmode which = ios_base::out | ios_base::in);
template<class SAlloc>
  basic_stringstream(
    const basic_string<charT, traits, SAlloc>& s, const Allocator& a)
    : basic_stringstream(s, ios_base::out | ios_base::in, a) {}
template<class SAlloc>
  basic_stringstream(
    const basic_string<charT, traits, SAlloc>& s,
    ios_base::openmode which, const Allocator& a);
template<class SAlloc>
  explicit basic_stringstream(
    const basic_string<charT, traits, SAlloc>& s,
    ios_base::openmode which = ios_base::out | ios_base::in);
template<typename T>
  explicit basic_stringstream(const T& t, ios_base::openmode which = ios_base::out | ios_base::in);
template<typename T>
  basic_stringstream(const T& t, const Allocator& a);
template<typename T>
  basic_stringstream(const T& t, ios_base::openmode which, const Allocator& a);
basic_stringstream(const basic_stringstream&) = delete;
basic_stringstream(basic_stringstream&& rhs);

...
// 31.8.5.4, members
basic_stringbuf<charT, traits, Allocator>* rdbuf() const;

basic_string<charT, traits, Allocator> str() const &;
template<class SAlloc>
  basic_string<charT,traits,SAlloc> str(const SAlloc& sa) const;
basic_string<charT, traits, Allocator> str() &&;
basic_string_view<charT, traits> view() const noexcept;

void str(const basic_string<charT, traits, Allocator>& s);
template<class SAlloc>
  void str(const basic_string<charT, traits, SAlloc>& s);
void str(basic_string<charT, traits, Allocator>&& s);
template<typename T>
  void str(const T& t);
```

[stringstream.cons]

In [stringstream.cons]:

```
      template<class SAlloc>
```

```
      explicit basic_stringstream(
        const basic_string<charT, traits, SAlloc>& s,
        ios_base::openmode which = ios_base::out | ios_base::in);
6     Constraints: is_same_v<SAlloc, Allocator> is false.
7     Effects: Initializes the base class with basic_iostream<charT, traits>(addressof(sb)) (31.7.4.7.2), and sb with basic_string-
      buf<charT, traits, Allocator>(s, which) (31.8.2.2).

    template<typename T>
      explicit basic_stringstream(const T& t, ios_base::openmode which = ios_base::out | ios_base::in);
8     Constraints:
(8.1) — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(8.2) — is_convertible_v<const T&, const charT*> is false.
9     Effects: Initializes the base class with basic_iostream<charT, traits>(addressof(sb)) (31.7.4.7.2) and sb with basic_string-
        buf<charT, traits, Allocator>(t, which) (31.8.2.2).

    template<typename T>
      basic_stringstream(const T& t, const Allocator& a);
10    Constraints:
(10.1) — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(10.2) — is_convertible_v<const T&, const charT*> is false.
11    Effects: Equivalent to basic_stringstream(t, ios_base::out | ios_base::in, a).

    template<typename T>
      basic_stringstream(const T& t, ios_base::openmode which, const Allocator& a);
12    Constraints:
(12.1) — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(12.2) — is_convertible_v<const T&, const charT*> is false.
13    Effects: Initializes the base class with basic_iostream<charT, traits>(addressof(sb)) (31.7.4.7.2) and sb with basic_string-
        buf<charT, traits, Allocator>(t, which, a) (31.8.2.2).

    basic_stringstream(basic_stringstream&& rhs);
```

[stringstream.members]

In [stringstream.members]:

```
    void str(basic_string<charT, traits, Allocator>&& s);
8     Effects: Equivalent to: rdbuf()->str(std::move(s));

    template<typename T>
      void str(const T& t);
9     Constraints:
(9.1) — is_convertible_v<const T&, basic_string_view<charT, traits>> is true and
(9.2) — is_convertible_v<const T&, const charT*> is false.
10    Effects: Equivalent to: rdbuf()->str(t);
```

Acknowledgements

Thanks to RISC Software GmbH for supporting this work. Thanks to Peter Kulczycki and Bernhard Manfred Gruber for proof reading and discussions.