Structured binding declaration as a condition

Zhihao Yuan

2024/6/26

Example 1

Symantec

```
if (auto [first, last] = parse(begin(), end()))
{
    // interpret [first, last) into a value
}
```

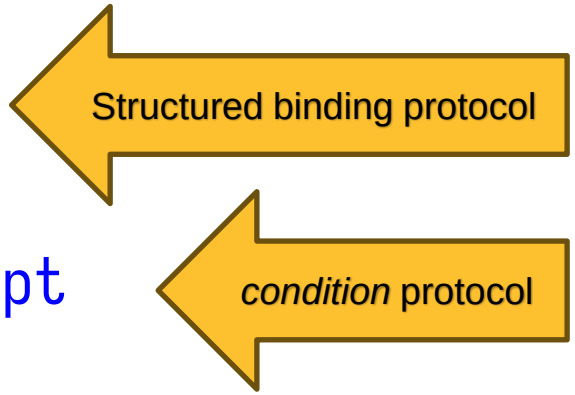
The return type implements two protocols

Symantec

```
struct parse_window
{
    char const *first, *last;

    explicit operator bool() const noexcept
    {
        return first != last;
    }
};

parse_window parse(char const*, char const*);
```



Without the paper

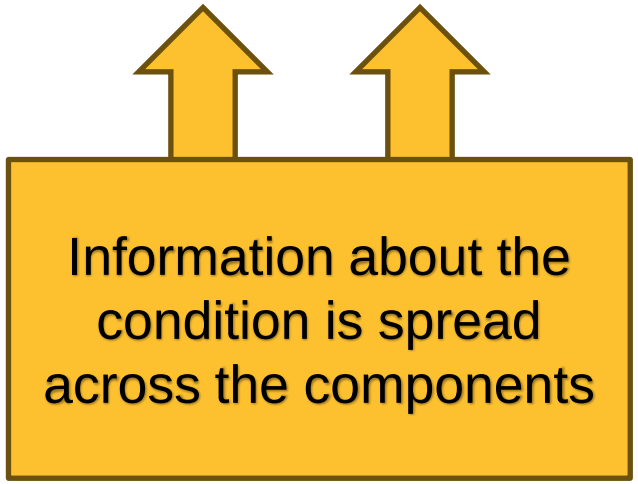
Symantec

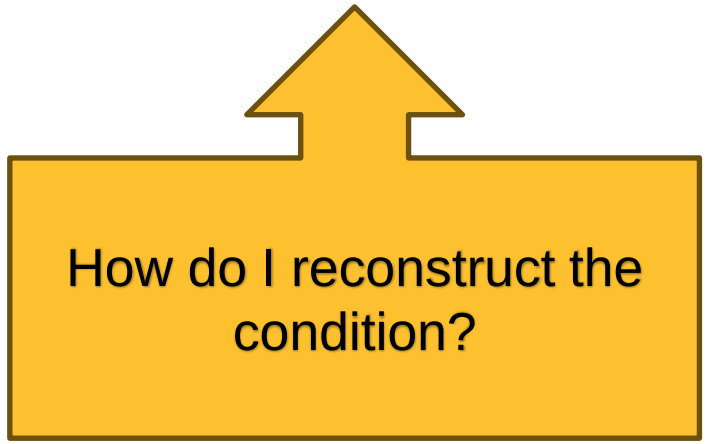
```
if (auto [first, last] = parse(begin(), end()); first != last)
{
    // interpret [first, last) into a value
}
```

Problem solved in example 1

Symantec

```
if (auto [first, last] = parse(begin(), end()); first != last)
{
```





Destructural & testable

Symantec

```
if (auto [first, last] = parse(begin(), end()))
{
    // interpret [first, last) into a value
}
```

Symantec

```
auto e = parse(begin(), end());
bool t(e.operator bool());
if (t)
{
```

```
auto e = parse(begin(), end());
using T1 = decltype((e.first));
using T2 = decltype((e.last));
T1 first = e.first;
T2 last = e.last;
```

Symantec

decision variable

```
auto e = parse(begin(), end());
using T1 = decltype((e.first));
using T2 = decltype((e.last));
bool t(e.operator bool());
T1 first = e.first;
T2 last = e.last;
if (t)
{
```

The formulation allows moving get()

Symantec

```
std::generator<int> f()
{
    co_yield 1;
    co_yield 2;
}

    if (auto g = f(); auto [a, b] = std::ranges::subrange{g})
    {
        // ok
    }
```

Move-only ranges

Symantec

```
template<std::size_t N, class I, class S, std::ranges::subrange_kind K>
    requires (N < 2)
constexpr auto get(std::ranges::subrange<I, S, K>&& r)
{
    if constexpr (N == 0)
        return r.begin(); // may perform move construction
    else
        return r.end();
}
```

Operator bool usable only if range isn't moved-out

Symantec

std::ranges::view_interface<D>::operator bool

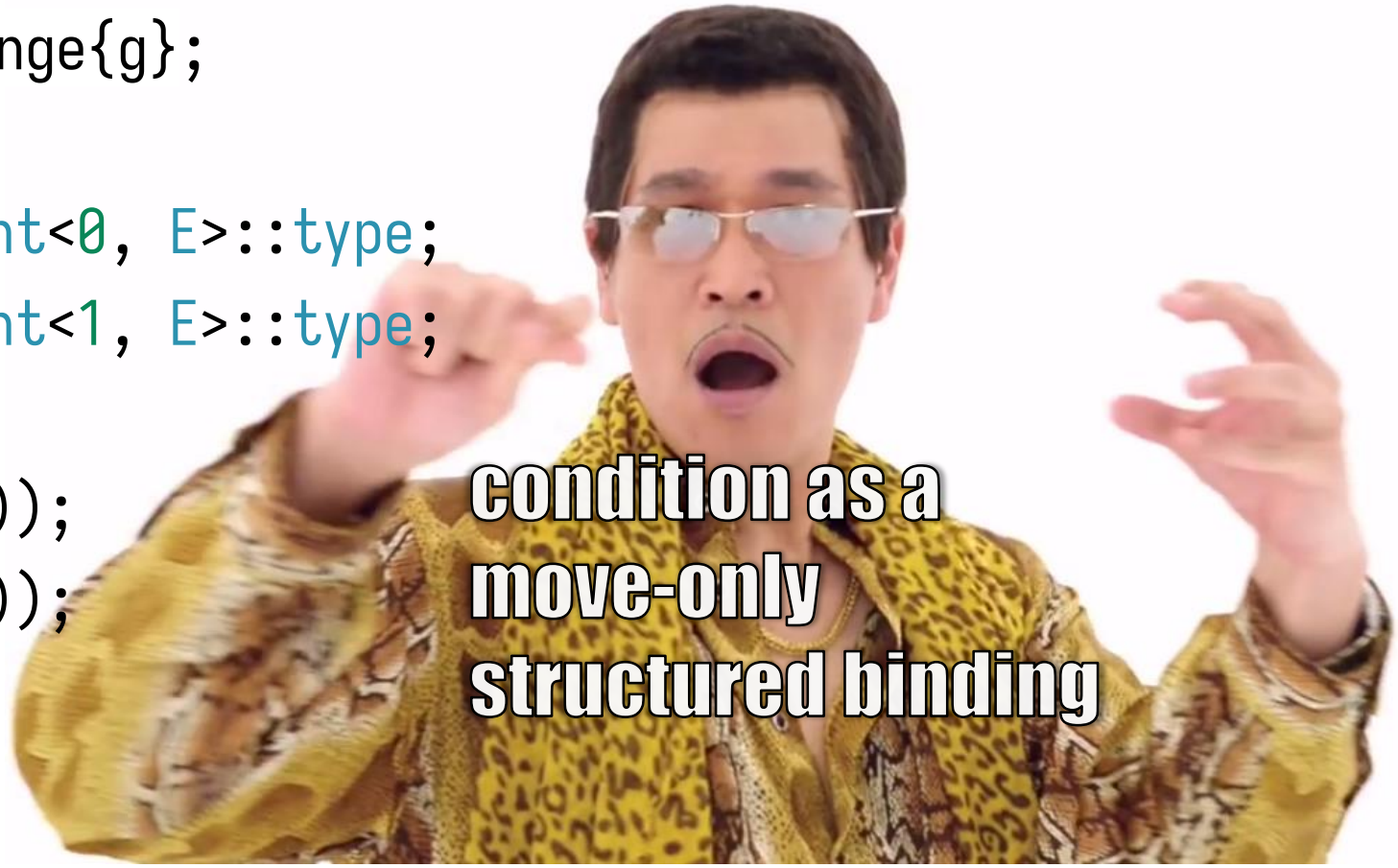
```
constexpr explicit operator bool() requires /* see below */;          (1)  (since C++20)
constexpr explicit operator bool() const requires /* see below */;    (2)  (since C++20)
```

The default implementation of `operator bool` member function checks whether the view is non-empty. It makes the derived type contextually convertible to `bool`.

1) Let `derived` be `static_cast<D&>(*this)`. The expression in the requires-clause is equal to `requires { ranges::empty(derived); }`, and the function body is equivalent to `return !ranges::empty(derived);`.
2) Same as (1), except that `derived` is `static_cast<const D&>(*this)`.

```
auto e = parse(begin(), end());
using T1 = decltype((e.first));
using T2 = decltype((e.last));
bool t(e.operator bool());
T1 first = e.first;
T2 last = e.last;
if (t)
{
```

```
auto e = std::ranges::subrange{g};
using E = decltype(e);
using T1 = std::tuple_element<0, E>::type;
using T2 = std::tuple_element<1, E>::type;
bool t(e.operator bool());
T1&& a = get<0>(std::move(e));
T2&& b = get<1>(std::move(e));
if (t)
{
```



Where can I use this?

Symantec

- if constexpr$_{opt}$ (*init-statement*$_{opt}$ *condition*) *statement* else *statement*
- switch (*init-statement*$_{opt}$ *condition*) *statement*
- while (*condition*) *statement*
- for (*init-statement condition*$_{opt}$; *expression*$_{opt}$) *statement*

Example 2

Symantec

```
if (auto [ptr, ec] = std::to_chars(p, last, 42))
{
    // okay to proceed
}
else
{
    // handle errors
}
```

Without this paper in C++26

Symantec

```
if (auto result = std::to_chars(p, last, 42))
{
    auto [ptr, _] = result;
    // okay to proceed
}
else
{
    auto [ptr, ec] = result;
    // handle errors
}
```

...or this

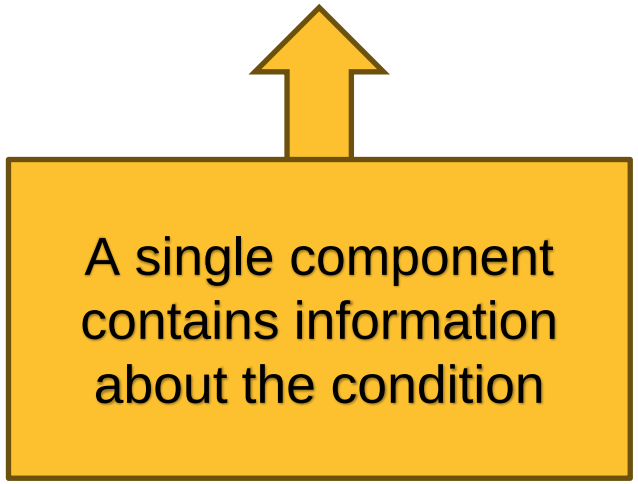
Symantec

```
if (auto [ptr, ec] = std::to_chars(p, last, 42); ec != std::errc{})
{
    // okay to proceed
}
else
{
    // handle errors
}
```

Problem solved in example 2

Symantec

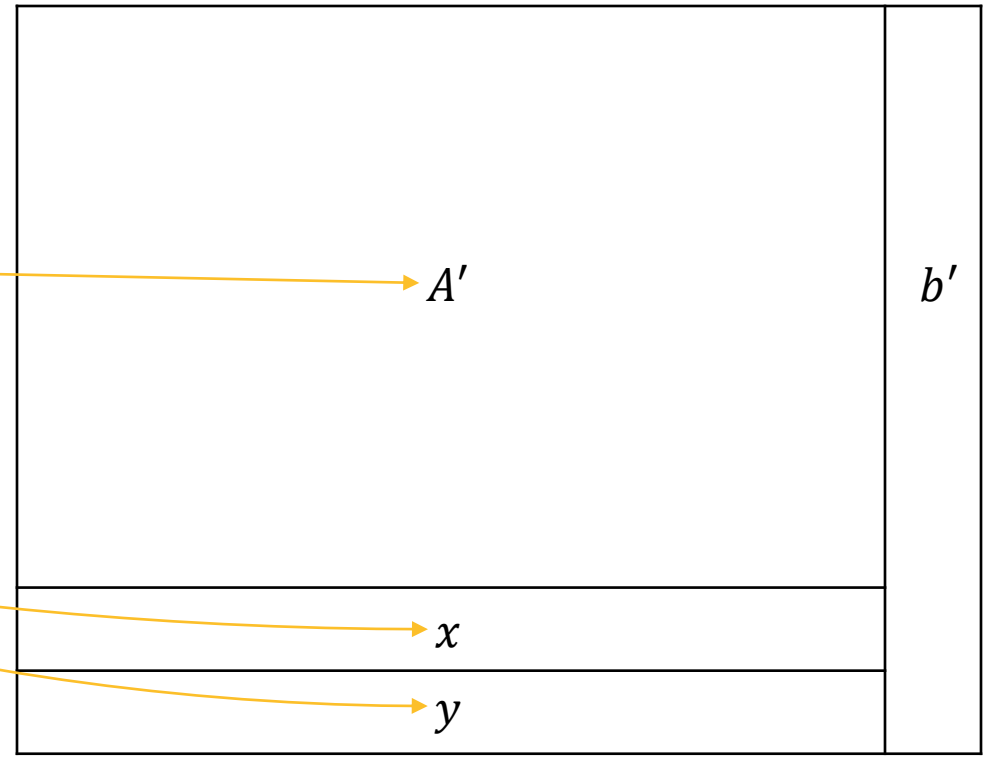
```
if (auto [ptr, ec] = std::to_chars(p, last, 42); ec != std::errc{})
{
```



Example 3

iterative solver

```
struct tableau
{
    matrix Ap;
    vector bp, x, y;
};
```



Example 3

iterative solver

Symantec

```
class tableau
{
    matrix Ap;
    vector bp, x, y;
    bool reached_optimal_;
};

// more code required
```

```
class Solver
{
  public:
    tableau solve();
    bool is_optimal(vector const&);
};
```

With this paper

Symantec

```
for (;;)
{
    // ...
    if (auto [Ap, bp, x, y] = solve())
        break;
}
```

Problem solved in example 3

Symantec

```
for (;;)
{
    // ...
    auto [Ap, bp, x, y] = solve();

    if (is_optimal(x))
        break;
}
```

Reconstructing information from the components is not zero-cost

Example 4

Symantec

```
if (auto [city, state, zip] = ctre2::match<"(\\w+), (\\w+) (\\d+)">(s))
{
    return location{ city, state, zip };
}
```

Without this paper

Symantec

```
if (auto [all, city, state, zip] =
        ctre::match<"(\\w+), (\\w+) (\\d+)">(s); all)
{
    return location{ city, state, zip };
}
```

Problem solved in example 4

Symantec

```
if (auto [all, city, state, zip] =
        ctre::match<"(\\w+), (\\w+) (\\d+)">(s); all)
{
```

All components but one have similar roles

Alternative solution *with* this paper in C++26

Symantec

```
if (auto [_, city, state, zip] =
        ctre::match<"(\\w+), (\\w+) (\\d+)">(s))
{
```

Implementation Experience

R1 is implemented in Clang

Symantec

C++

```
9
10  struct format_status
11  {
12      format_errc ec;
13      char *bp;
14
15      explicit operator bool() const noexcept
16      {
17          return ec == format_errc::no_error;
18      }
19  };
20
21  format_status
22
23  int main()
24  {
25      if (auto [ok, ptr] = readint())
26      {
27          printf("stopped at %p\n", ptr);
28      }
29  }
```

warning: ISO C++17 does not permit structured binding declaration in a condition [-Wbinding-in-condition] x86-64 clang (trunk) #1

View Problem (Alt+F8) No quick fixes available

x86-64 clang (trunk) -Wall -std=c++17

```
1   main: # @main
2     push rax
3     call readint()@PLT
4     test eax, eax
5     jne .LBB0_2
6     lea rdi, [rip + .L.str]
7     mov rsi, rdx
8     xor eax, eax
9     call printf@PLT
10  .LBB0_2:
11    xor eax, eax
12    pop rcx
13    ret
14  .L.str:
15    .asciz "stopped at %p\n"
```

R2 semantics is not in the extension

Symantec

C++ source #1

C++

```
1  #include <generator>
2  #include <ranges>
3
4  std::generator<int> f() {
5      co_yield 1;
6      co_yield 2;
7  }
8
9  int main() {
10     if (auto g = f();
11         auto [b, e] = std::ranges::subrange{g}) {
12         return 0;
13     }
14 }
```

Output of x86-64 clang (trunk) (Compiler #1)

Wrap lines | Select all

```
<source>:11:14: warning: ISO C++17 does not permit structured
binding declaration in a condition [-Wbinding-in-condition]
   11 |         auto [b, e] = std::ranges::subrange{g}) {
      |              ^~~~~~
1 warning generated.
ASM generation compiler returned: 0
<source>:11:14: warning: ISO C++17 does not permit structured
binding declaration in a condition [-Wbinding-in-condition]
   11 |         auto [b, e] = std::ranges::subrange{g}) {
      |              ^~~~~~
1 warning generated.
Execution build compiler returned: 0
Program returned: 139
 Program terminated with signal: SIGSEGV
```

Thank you