31

```cpp
template<typename F, send... Args>
safe_thread (F&& f, Args&&... args)
    : thread (std::forward<F> (f), std::forward<Args> (args)...)
{
    static_assert (send<F>);
}
```

```
template<typename T>
struct is_send : std::bool_constant<
        (! (std::is_lvalue_reference_v<T>
            || std::is_pointer_v<std::remove_extent_t<T>>
            || is_lambda_v<T>))
         &&
           (std::is_move_constructible_v<T>
            || (is_function_pointer_v<std::decay_t<T>>
                && ! std::is_member_function_pointer_v<T>))>
{};

template<typename T>
concept send = is_send<T>::value;
```

```cpp
static_assert(is_send_v<const int>);
static_assert(is_send_v<int>);
static_assert(is_send_v<int&&>);
static_assert(is_send_v<int>);

static_assert(! is_send_v<int&>);
static_assert(! is_send_v<int*&>);
static_assert(! is_send_v<const int&>);
static_assert(! is_send_v<const int*&>);
static_assert(! is_send_v<std::string&>);
static_assert(! is_send_v<const std::string&>);
static_assert(! is_send_v<std::string*&>);
static_assert(! is_send_v<const std::string*&>);
```

Send in C++: *Moved between threads*

Send in C++: *Moved between threads*

```cpp
template<typename F, send... Args>
safe_thread (F&& f, Args&&... args)
    : thread (std::forward<F> (f), std::forward<Args> (args)...)
{
    static_assert (send<F>);
}
```

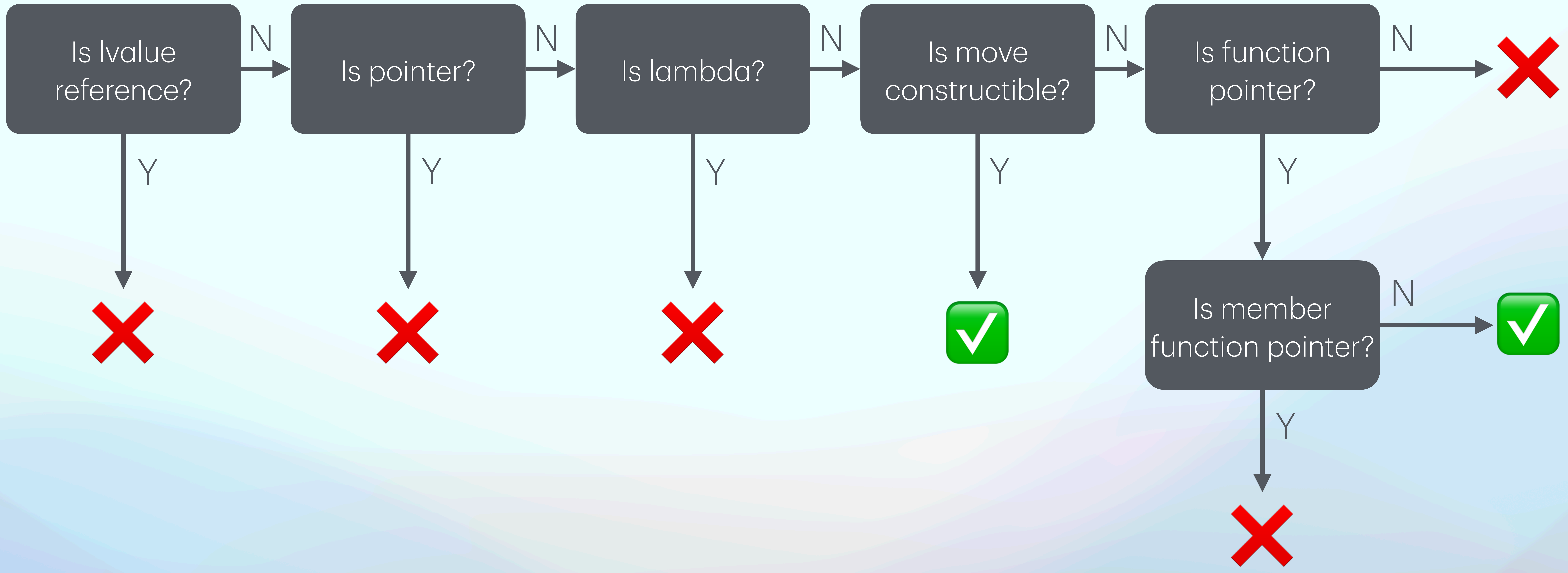
```cpp
template<typename T>
struct is_send : std::bool_constant<
       (! (std::is_lvalue_reference_v<T>
           || std::is_pointer_v<std::remove_extent_t<T>>
           || is_lambda_v<T>))
        &&
          (std::is_move_constructible_v<T>
           || (is_function_pointer_v<std::decay_t<T>>
               && ! std::is_member_function_pointer_v<T>))>
{};

template<typename T>
concept send = is_send<T>::value;
```

```cpp
static_assert(is_send_v<const int>);
static_assert(is_send_v<int>);
static_assert(is_send_v<int&&>);
static_assert(is_send_v<int>);

static_assert(! is_send_v<int&>);
static_assert(! is_send_v<int*&>);
static_assert(! is_send_v<const int&>);
static_assert(! is_send_v<const int*&>);
static_assert(! is_send_v<std::string&>);
static_assert(! is_send_v<const std::string&>);
static_assert(! is_send_v<std::string*&>);
static_assert(! is_send_v<const std::string*&>);
```

Is T `send`?


Is lvalue reference?
N
Y
✗
Is pointer?
N
Y
✗
Is lambda?
N
Y
✗
Is move constructible?
N
Y
✅
Is function pointer?
N
✗
Y
Is member function pointer?
N
✅
Y
✗