How To Initialize **x** from expression **y**

Howard Hinnant

Meeting C++
2019-11-16

How To Initialize **x** from expression **y**

- You think that this would be simple, but modern C++ gives us so many options.
- Options are good when they allow you to fine tune your code.
- You just have to know how to use them.

How To Initialize **x** from expression **y**

Should x and y have the same cv-unqualified type?

- Here are several of those options:
- Let's start by classifying the use cases.

```
auto x = y;
```

```
auto& x = y;
```

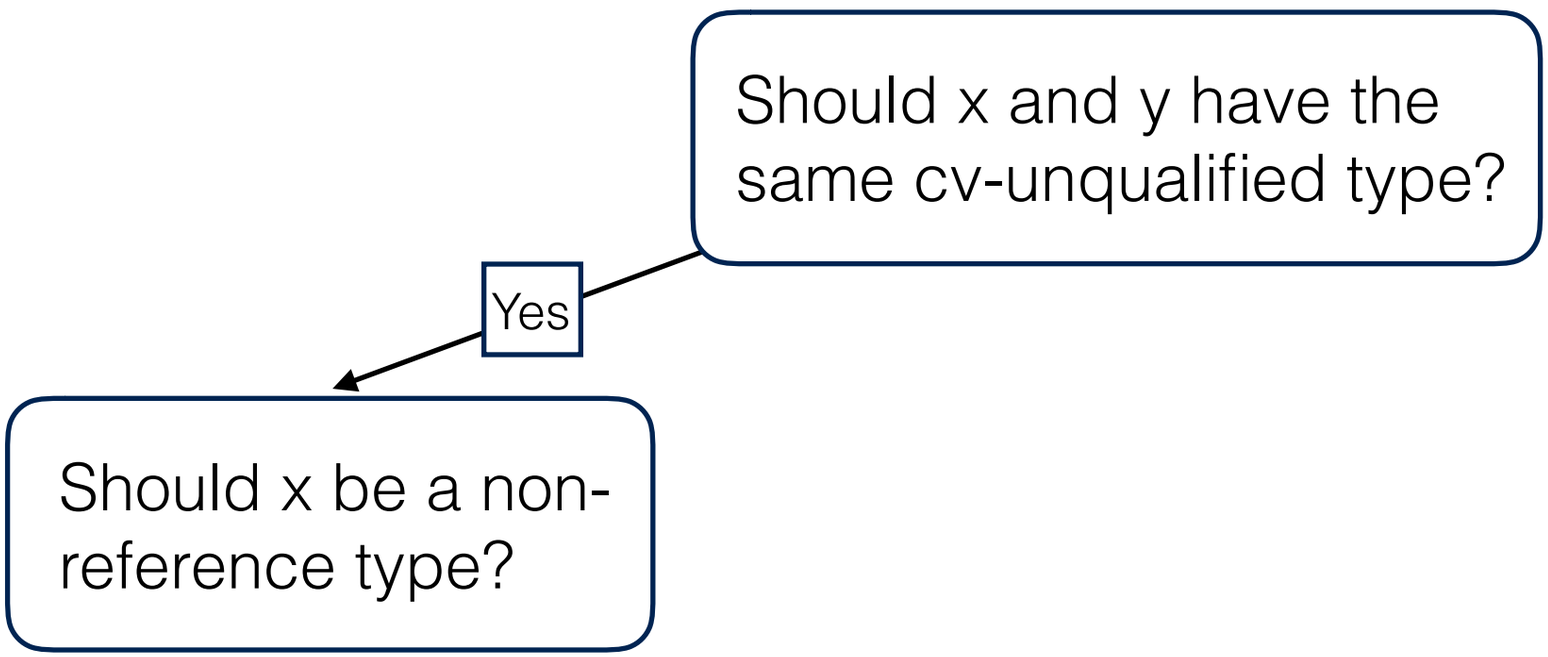
```
auto&& x = y;
```

```
X x = y;
```

```
auto x = X{y};
```

```
auto x = X(y);
```

How To Initialize **x** from expression **y**




```
auto x = y;
```

```
auto& x = y;
```

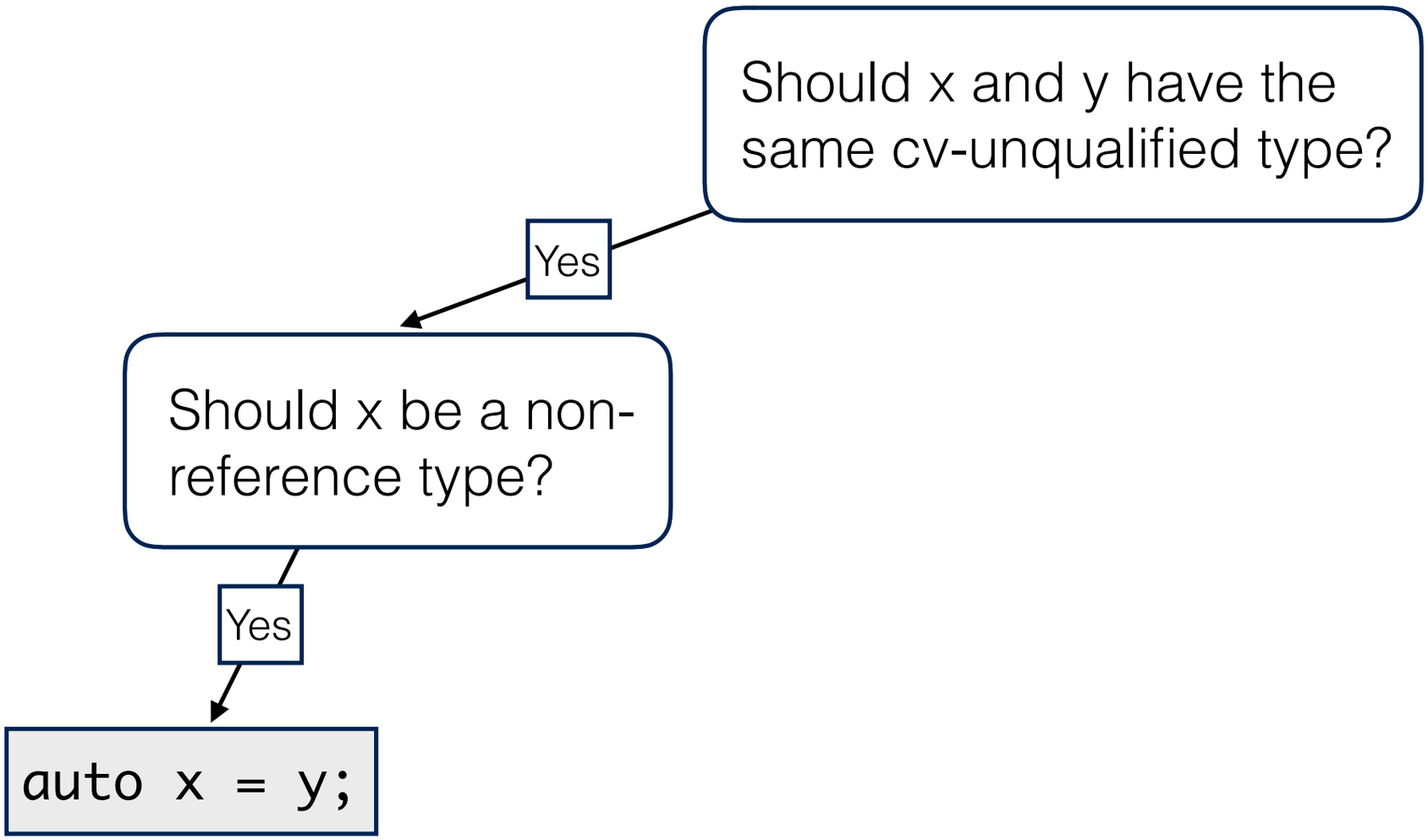
```
auto&& x = y;
```

```
X x = y;
```

```
auto x = X{y};
```

```
auto x = X(y);
```

How To Initialize **x** from expression **y**




```
auto& x = y;
```

```
auto&& x = y;
```

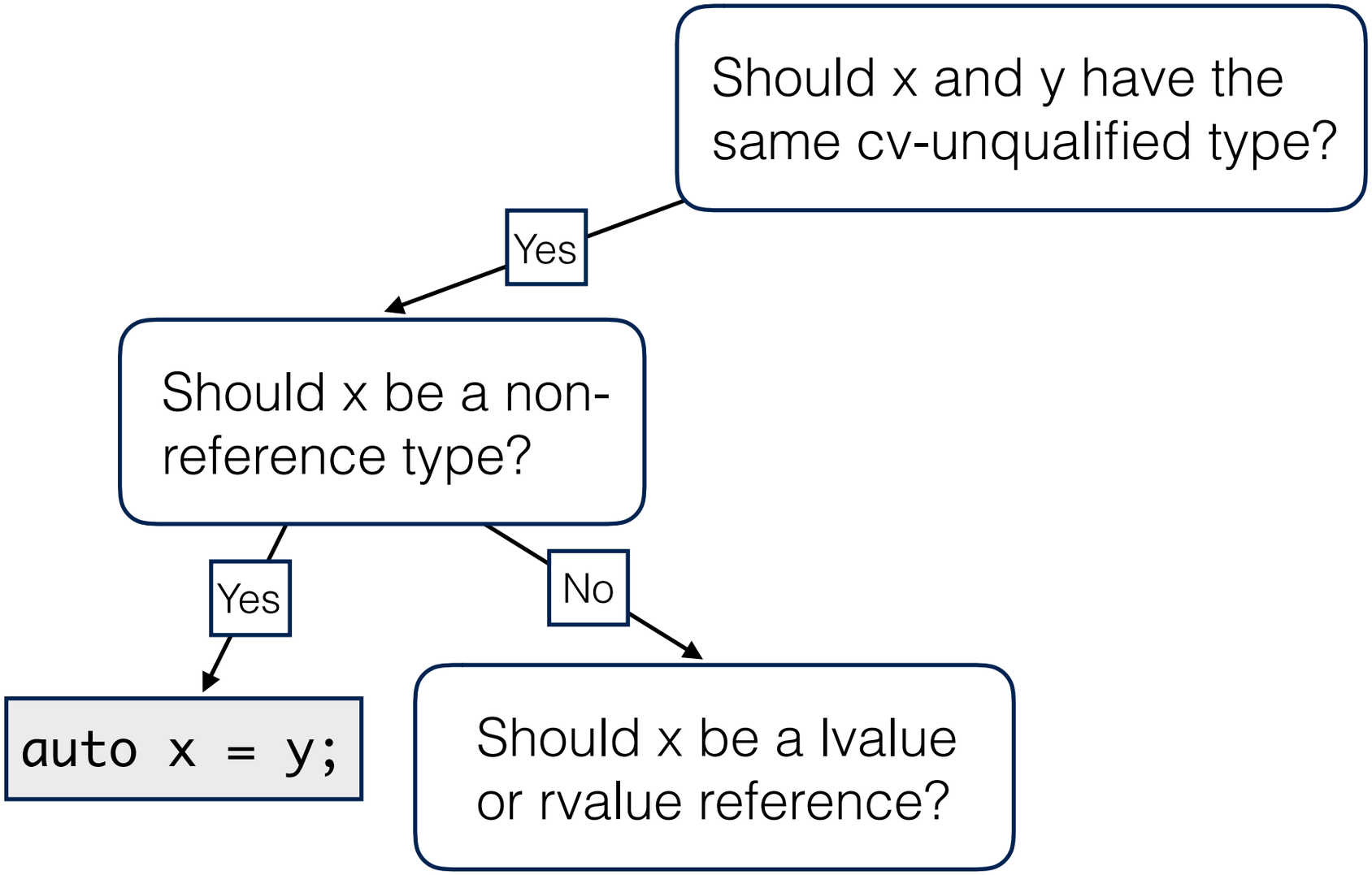
```
X x = y;
```

```
auto x = X{y};
```

```
auto x = X(y);
```

Note: The copy constructor nor the move constructor should *ever* be marked `explicit`, else this simple syntax will fail (for no good reason).

How To Initialize **x** from expression **y**




```
auto& x = y;
```

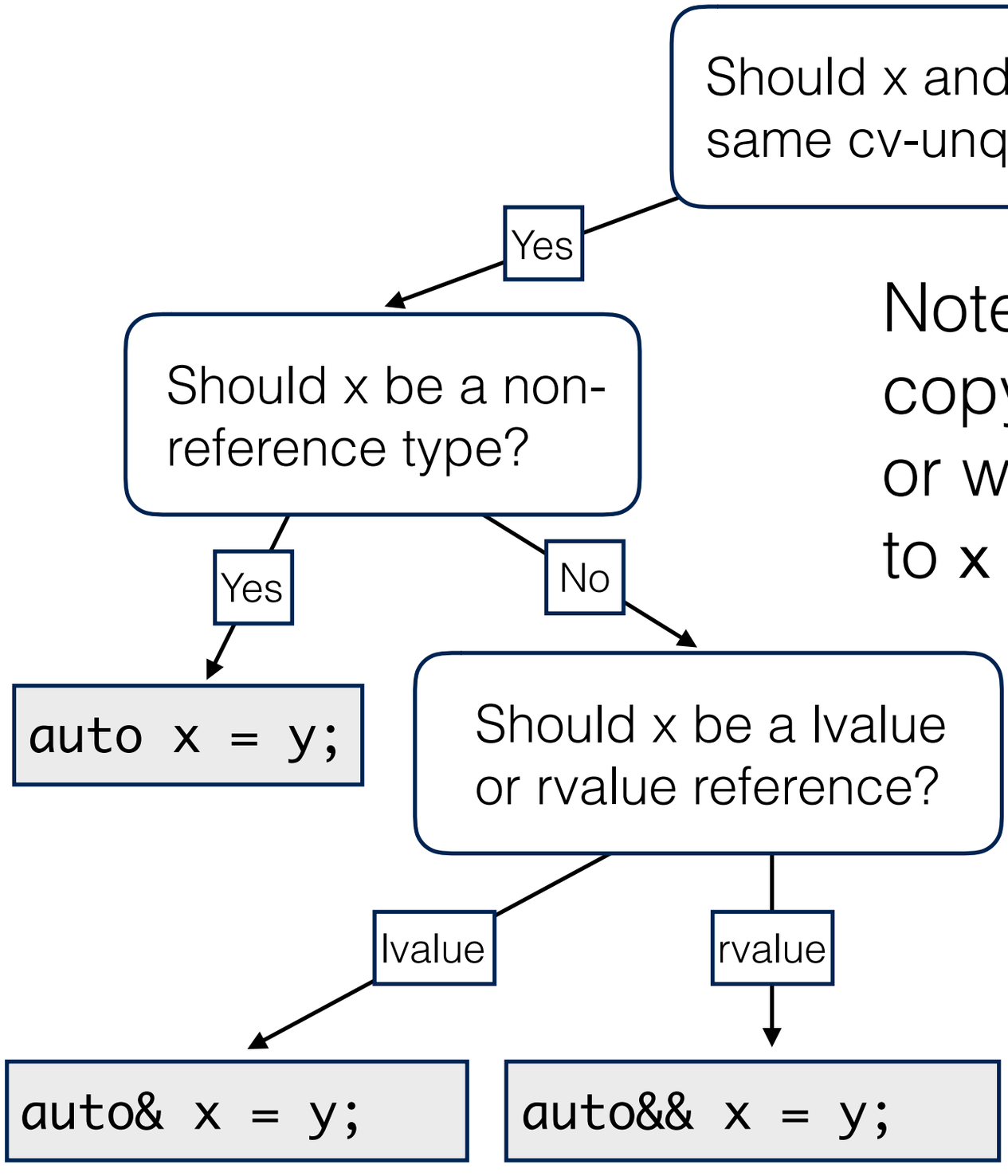
```
auto&& x = y;
```

```
X x = y;
```

```
auto x = X{y};
```

```
auto x = X(y);
```

How To Initialize **x** from expression **y**



Note: These are handy when a copy or move should be avoided, or when you want modifications to `x` to write through to `y`.

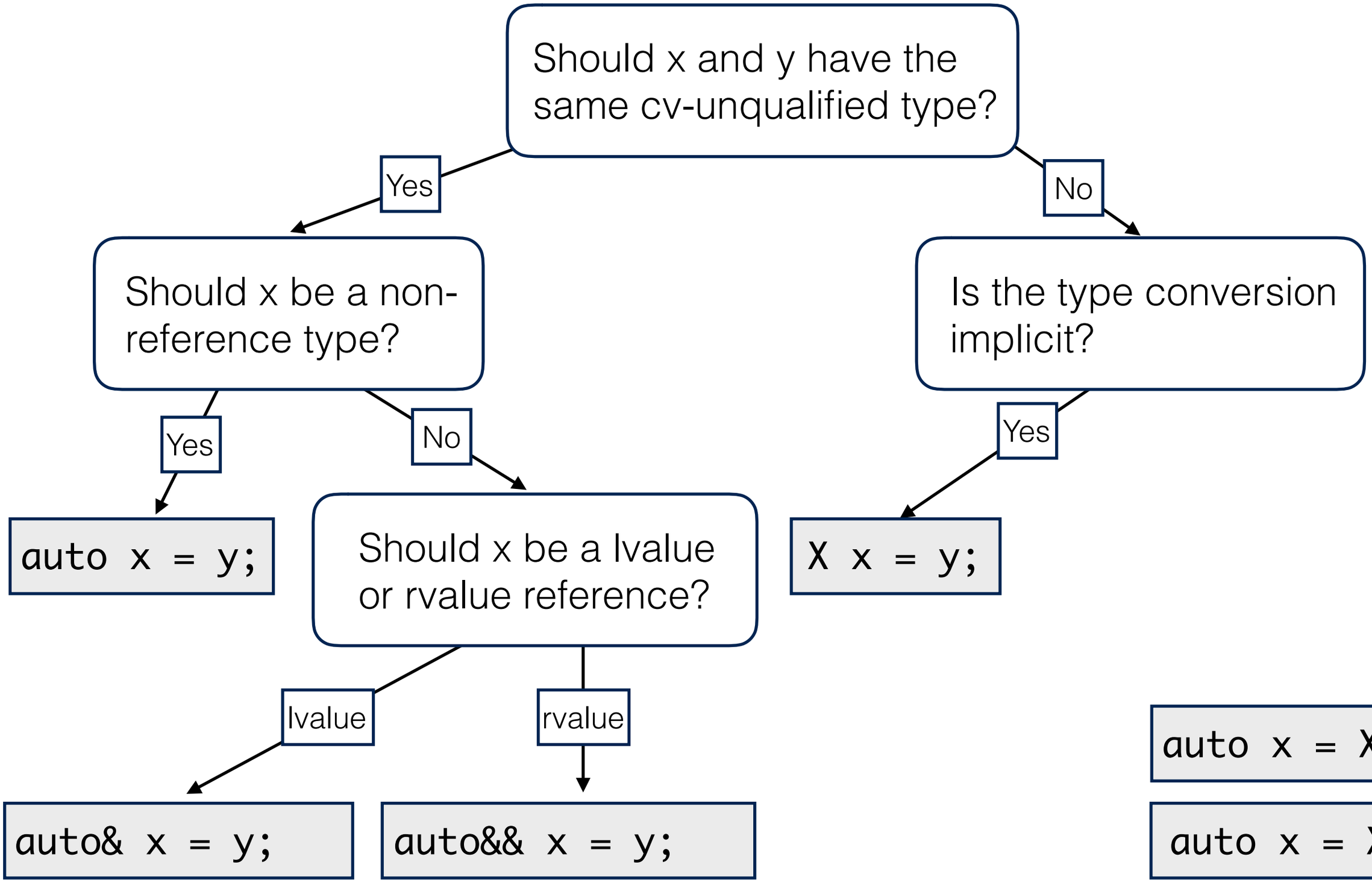
```
for (auto& x : y)
```

```
X x = y;
```

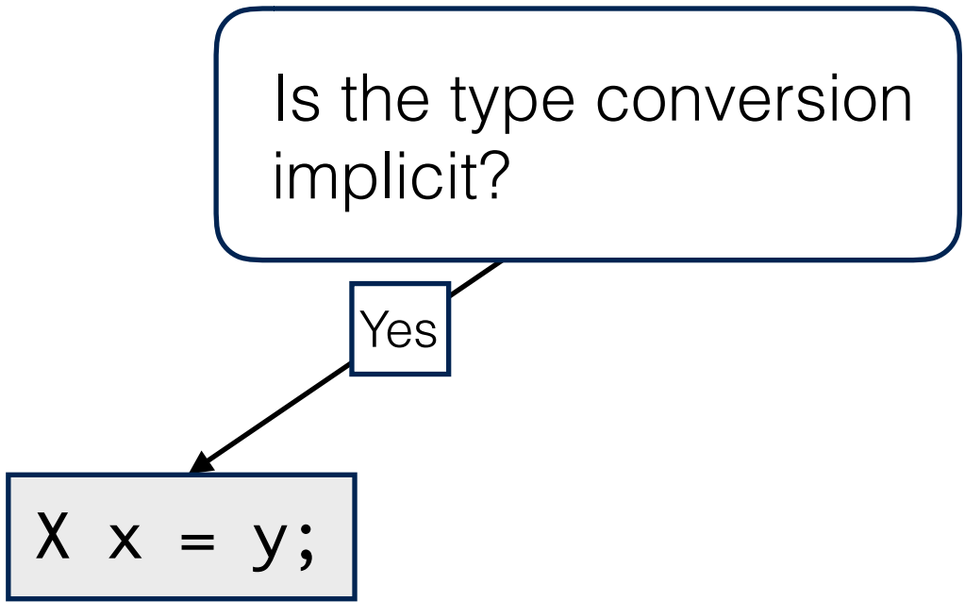
```
auto x = X{y};
```

```
auto x = X(y);
```

How To Initialize **x** from expression **y**



How To Initialize **x** from expression **y**



Prefer implicit conversions?!

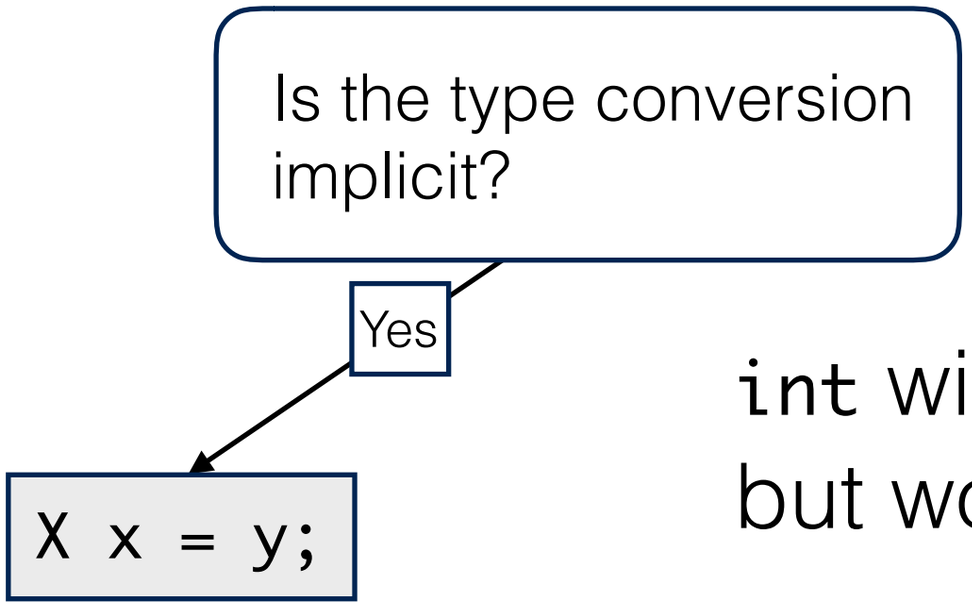
```
template <class Duration1, class Duration2>
auto
avg_nanoseconds(Duration1 d1, Duration2 d2)
{
    using namespace std::chrono;
    auto ns = nanoseconds{d1 + d2};
    return ns/2;
}

auto x = avg_nanoseconds(2us, 1ms);  // 501000ns
```

Explicit conversion

Good!

How To Initialize **x** from expression **y**



Prefer implicit conversions?!

`int` will *explicitly* convert to `nanoseconds`,
but won't *implicitly* convert to `nanoseconds`.

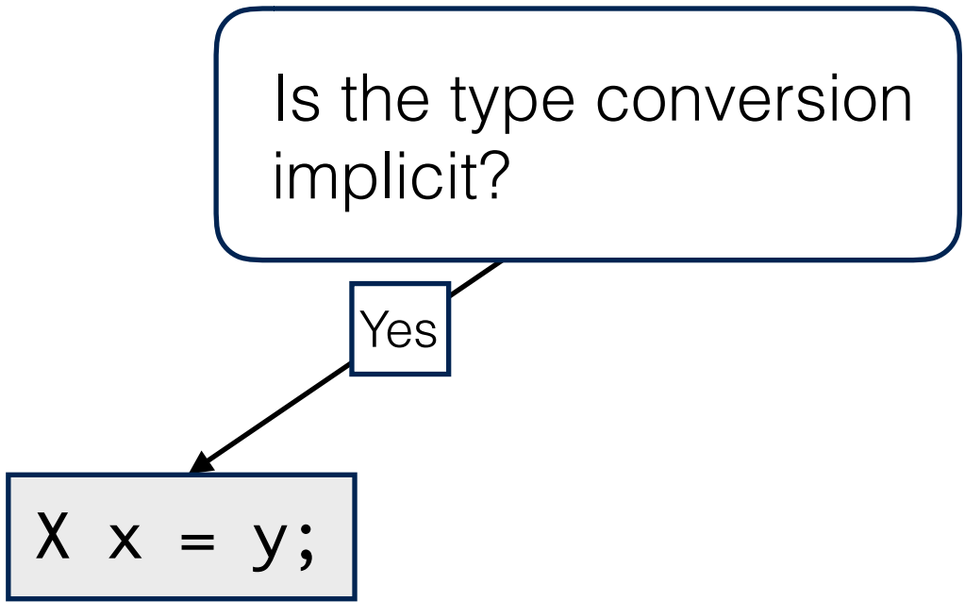
```
template <class Duration1, class Duration2>
auto
avg_nanoseconds(Duration1 d1, Duration2 d2)
{
    using namespace std::chrono;
    auto ns = nanoseconds{d1 + d2};
    return ns/2;
}

auto x = avg_nanoseconds(2, 1);      // 1ns
```

Explicit conversion

Oops! Run-time error!

How To Initialize **x** from expression **y**



Prefer implicit conversions?!

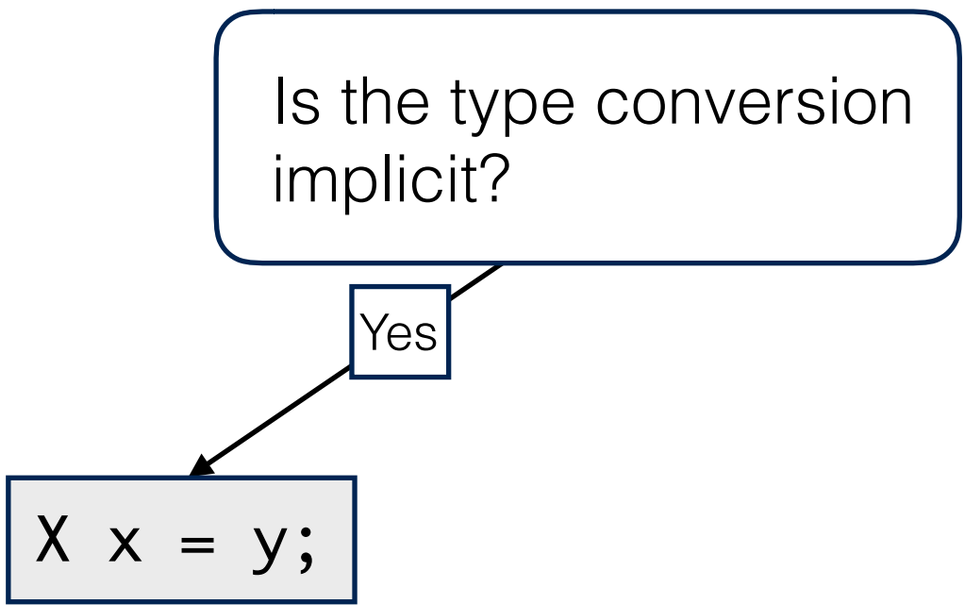
```
template <class Duration1, class Duration2>
auto
avg_nanoseconds(Duration1 d1, Duration2 d2)
{
    using namespace std::chrono;
    nanoseconds ns = d1 + d2;
    return ns/2;
}

auto x = avg_nanoseconds(2us, 1ms);  // 501000ns
```

Implicit conversion

Still good!

How To Initialize **x** from expression **y**



Prefer implicit conversions?!

Safest choice!

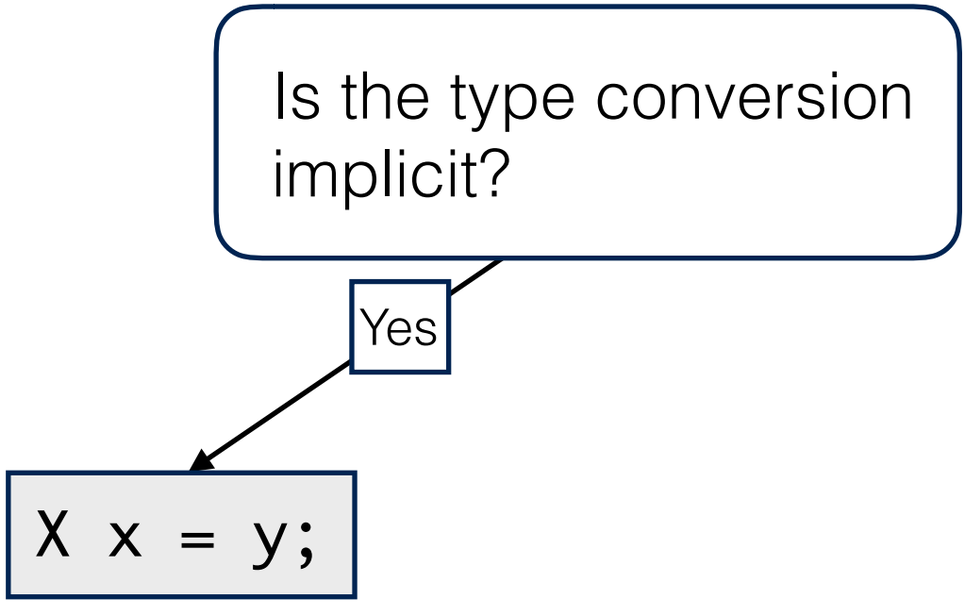
```
template <class Duration1, class Duration2>
auto
avg_nanoseconds(Duration1 d1, Duration2 d2)
{
    using namespace std::chrono;
    nanoseconds ns = d1 + d2;
    return ns/2;
}

auto x = avg_nanoseconds(2, 1);
```

Implicit conversion

```
error: no viable conversion from 'int' to 'nanoseconds'
    nanoseconds ns = d1 + d2;
                 ^    ~~~~~~
```

How To Initialize **x** from expression **y**



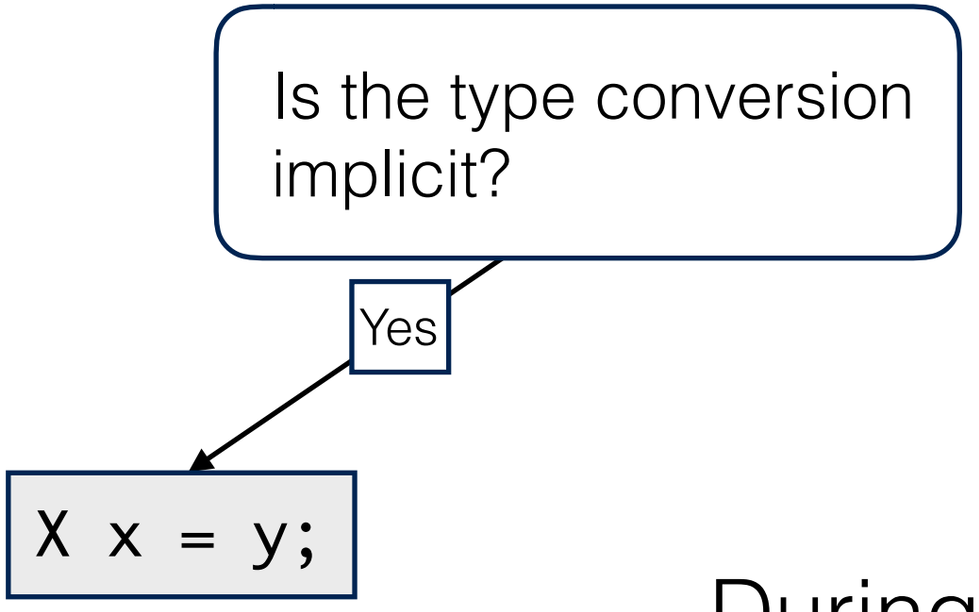
Prefer implicit conversions?!

This is not just a <chrono> issue!

```
auto
f(shared_ptr<Derived> p)
{
    // lots of code (too much really)…
    auto bp = shared_ptr<Base>{p};
    // more code…
}
```

Explicit conversion

How To Initialize **x** from expression **y**



Prefer implicit conversions?!

This is not just a <chrono> issue!

During refactor:

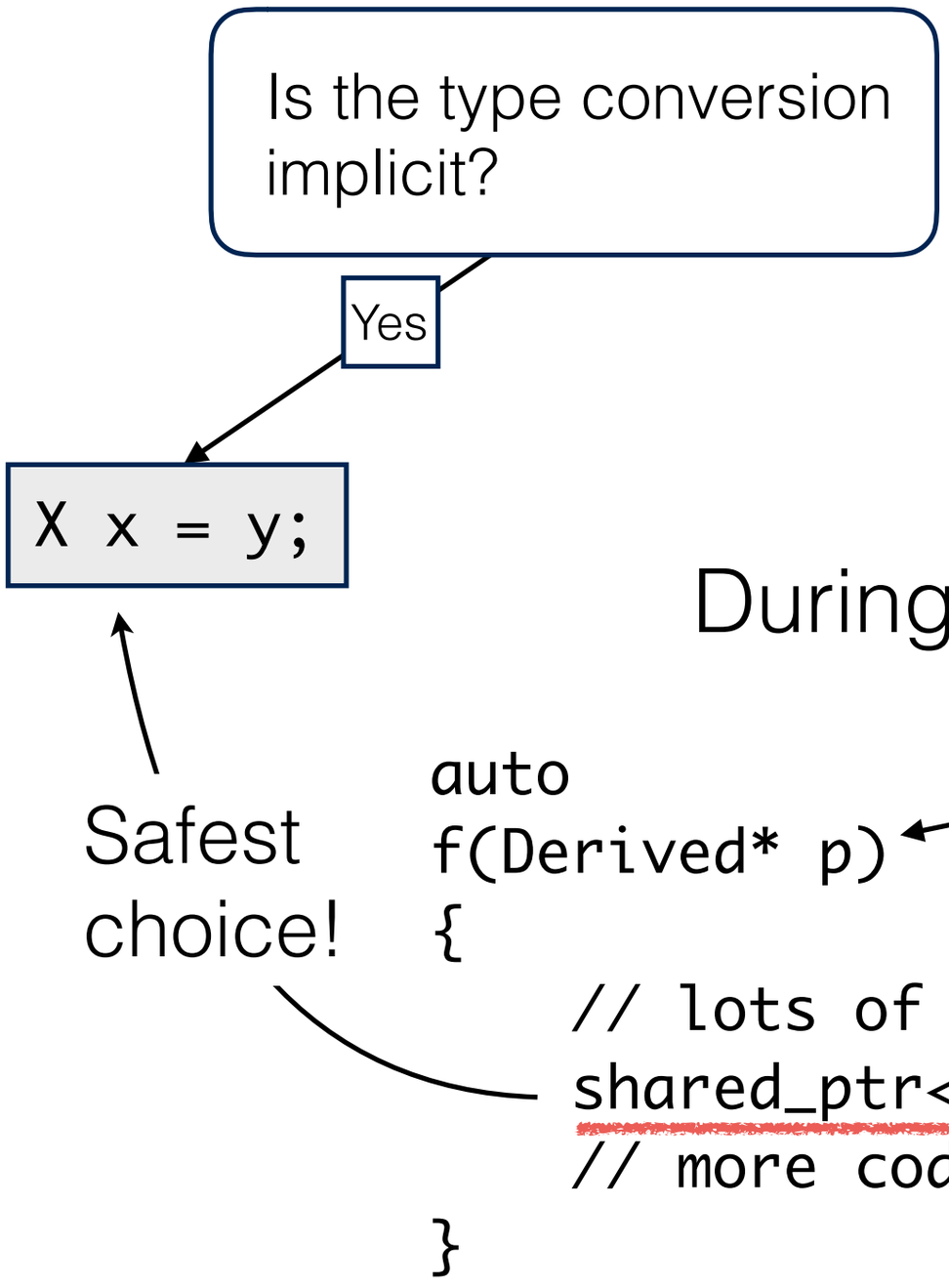
```
auto
f(Derived* p)
{
    // lots of code (too much really)…
    auto bp = shared_ptr<Base>{p};
    // more code…
}
```

was: `shared_ptr<Derived>`

Explicit conversion

Run-time error!

How To Initialize **x** from expression **y**



Prefer implicit conversions?!

This is not just a <chrono> issue!

During refactor:

was: `shared_ptr<Derived>`

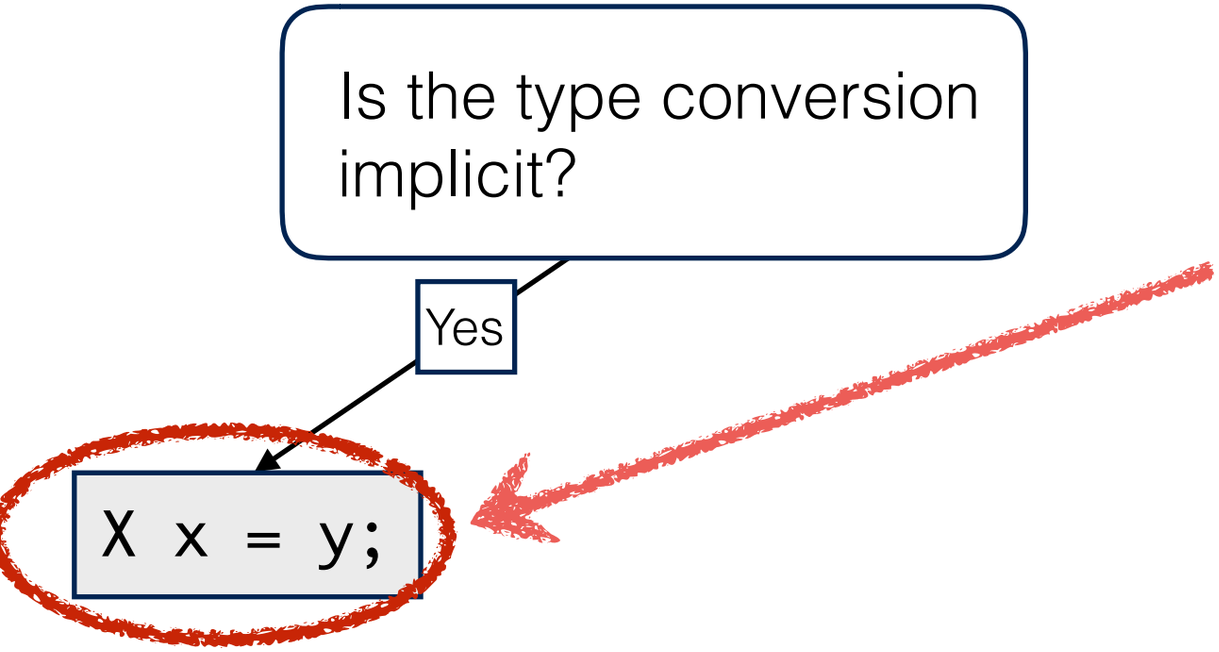
```
auto
f(Derived* p)
{
    // lots of code (too much really)…
    shared_ptr<Base> bp = p;
    // more code…
}
```

Implicit conversion

Compile-time error!

Fix with: `Base* bp = p;`

How To Initialize **x** from expression **y**

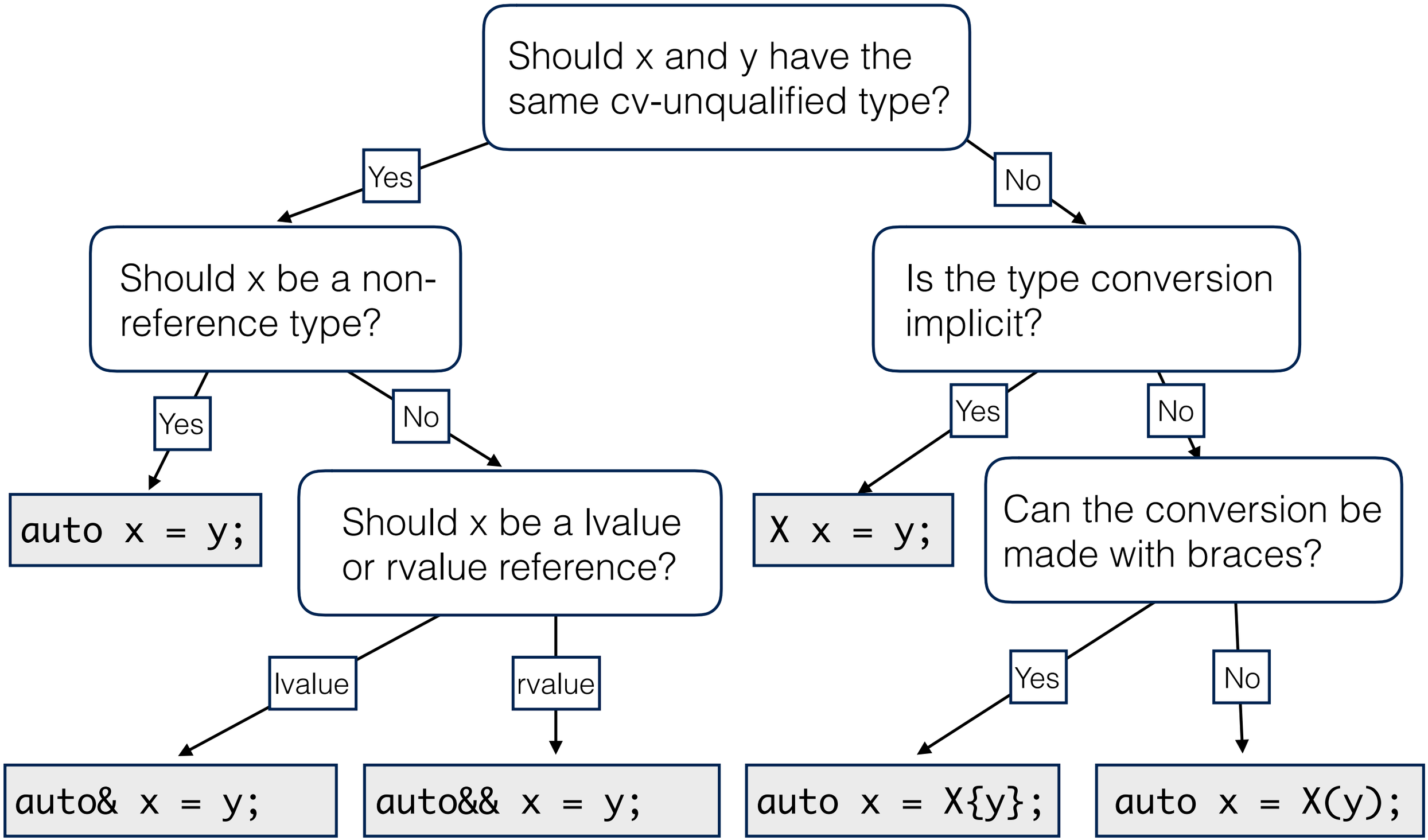


Prefer implicit conversions?!

Yes, for clients!
No, for type authors.
The optimum lives between these two interests.

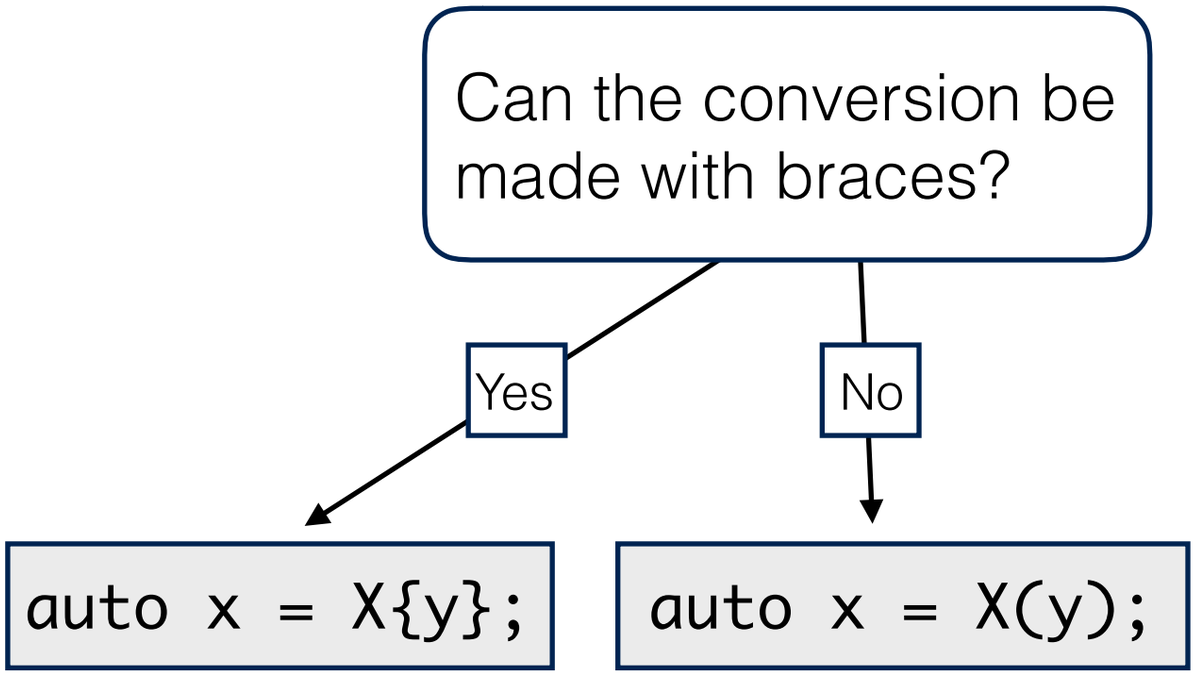
- Clients should prefer implicit conversions because these are the conversions the type author considers the safest.
- Type authors should use `explicit` for all conversions when the meaning of the two types is drastically different.

How To Initialize **x** from expression **y**



Add const (and/or volatile) as appropriate.

How To Initialize **x** from expression **y**

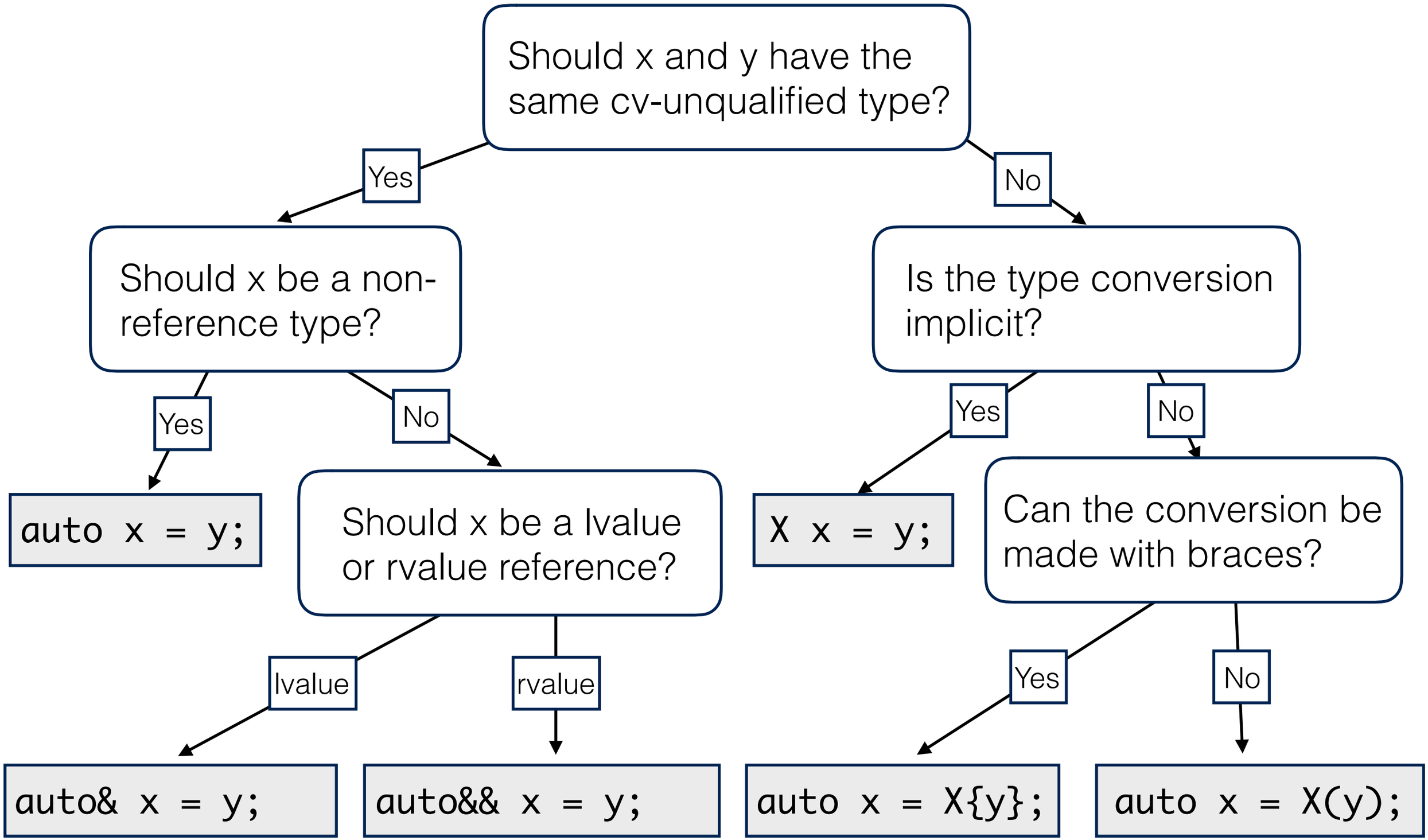


For example:

```
auto v1 = vector<int>{3};  // v1 = {3}

auto v2 = vector<int>(3);  // v3 = {0, 0, 0}
```

How To Initialize **x** from expression **y**



Add const (and/or volatile) as appropriate.


That's all Folks!