Fixing Two-Phase Initialization

Andreas Weis

BMW AG

CppCon 2018

`-fno-exceptions`

The Problem

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
public:
  Foo(Arg n_arg)
  :m_state(std::make_unique<InternalState>(n_arg))
  { }
};
```

The Problem

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
public:
  Foo(Arg n_arg)
  :m_state(std::make_unique<InternalState>(n_arg))
  { }
};
```

Two-Phase Initialisation

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
public:
  Foo() noexcept
  :m_state()
  { }
};
```

Two-Phase Initialisation

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
public:
  Foo() noexcept
  :m_state()
  { }









};
```

Two-Phase Initialisation

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
public:
  Foo() noexcept
  :m_state()
  { }

  std::error_code init(Arg n_arg) noexcept {
    m_state = make_unique_nothrow(n_arg);


  }
};
```

Two-Phase Initialisation

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
public:
  Foo() noexcept
  :m_state()
  { }

  std::error_code init(Arg n_arg) noexcept {
    m_state = make_unique_nothrow(n_arg);
    if(!m_state) { return { my_errc::error, my_category() }; }
    return std::error_code();
  }
};
```

- Objects in partial constructed state

A first attempt to fix this...

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
public:
  Foo() noexcept
  :m_state()
  { }

  std::error_code init(Arg n_arg) noexcept {
    m_state = make_unique_nothrow(n_arg);
    if(!m_state) { return { my_errc::error, my_category() }; }
    return std::error_code();
  }
};
```

A first attempt to fix this...

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;

  Foo() noexcept
  :m_state()
  { }
public:
  std::error_code init(Arg n_arg) noexcept {
    m_state = make_unique_nothrow(n_arg);
    if(!m_state) { return { my_errc::error, my_category() }; }
    return std::error_code();
  }
};
```

A first attempt to fix this...

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
  Foo() noexcept
  :m_state()
  { }
public:
  static expected<Foo> create(Arg n_arg) noexcept {




  }
};
```

A first attempt to fix this...

```
class Foo {
private:
  std::unique_ptr<InternalState> m_state;
  Foo() noexcept
  :m_state()
  { }
public:
  static expected<Foo> create(Arg n_arg) noexcept {
    Foo ret{};
    ret.m_state = make_unique_nothrow(n_arg);
    if(!ret.m_state) { return unexpected(my_errc::error); }
    return ret;
  }
};
```

- ~~Objects in partial constructed state~~ ✓

- ~~Objects in partial constructed state~~ ✓
- Non-idiomatic construction

Inverse Two-Phase Initialisation

```
static expected<Foo>
    create(Arg n_arg) noexcept
{
  Foo ret;
  ret.m_state = make_unique_nothrow(n_arg);
  if(!ret.m_state) { return unexpected(my_errc::error); }
  return ret;
}
```

Inverse Two-Phase Initialisation

```
static expected<construction_token>
    preconstruct(Arg n_arg) noexcept
{
  construction_token t;
  t.state = make_unique_nothrow(n_arg);
  if(!t.state) { return unexpected(my_errc::error); }
  return t;
}
```

Inverse Two-Phase Initialisation

```
static expected<construction_token>
    preconstruct(Arg n_arg) noexcept
{
  construction_token t;
  t.state = make_unique_nothrow(n_arg);
  if(!t.state) { return unexpected(my_errc::error); }
  return t;
}

Foo(construction_token&& t) noexcept
:m_state(std::move(t.state))
{ }
```

Inverse Two-Phase Initialisation (User's view)

```
expected<Foo::construction_token> t1 = Foo::preconstruct(args);
if(!t1.has_value()) { /* get out... */ }
Foo obj(std::move(*t1));
```

Inverse Two-Phase Initialisation (User's view)

```
expected<Foo::construction_token> t1 = Foo::preconstruct(args);
if(!t1.has_value()) { /* get out... */ }
Foo obj(std::move(*t1));

 // or
auto t2 = Foo::preconstruct(args);
auto obj_ptr = std::make_shared<Foo>(std::move(*t2));
```

Inverse Two-Phase Initialisation (User's view)

```
expected<Foo::construction_token> t1 = Foo::preconstruct(args);
if(!t1.has_value()) { /* get out... */ }
Foo obj(std::move(*t1));

 // or
auto t2 = Foo::preconstruct(args);
auto obj_ptr = std::make_shared<Foo>(std::move(*t2));

 // or
auto t3 = Foo::preconstruct(args);
std::vector<Foo> objects;
objects.emplace_back(std::move(*t3));
```

- ~~Objects in partial constructed state~~ ✓
- ~~Non-idiomatic construction~~ ✓

Inverse Two-Phase Initialisation

```
static expected<construction_token>
    preconstruct(Arg n_arg) noexcept
{
  construction_token t;
  t.state = make_unique_nothrow(n_arg);
  if(!t.state) { return unexpected(my_errc::error); }
  return t;
}

Foo(construction_token&& t) noexcept
:m_state(std::move(t.state))
{ }
```