

Monadic error handling in C++ template metaprograms

Ábel Sinkovics

ELTE, Hungary

Outline

- Introduction to template metaprogramming
- Error handling using an example
- Monads in template metaprogramming
- Summary

C++ template metaprogramming

- Erwin Unruh, 1994
- Code executed at compile-time
- Turing-complete
- Purely functional

- Concept checking
- Expression templates
- DSL embedding

C++ template metafunction

```
template <class T>  
struct makeConst  
{  
    typedef const T type;  
};
```

Argument list

Name

Body

C++ template metafunction

```
template <class T>  
struct makeConst  
{  
    typedef const T type;  
};
```

Argument list

Name

Body

C++ template metafunction

```
template <class T>  
struct makeConst  
{  
    typedef const T type;  
};
```

Argument list

Name

Body

makeConst<int>::type

C++ template metafunction

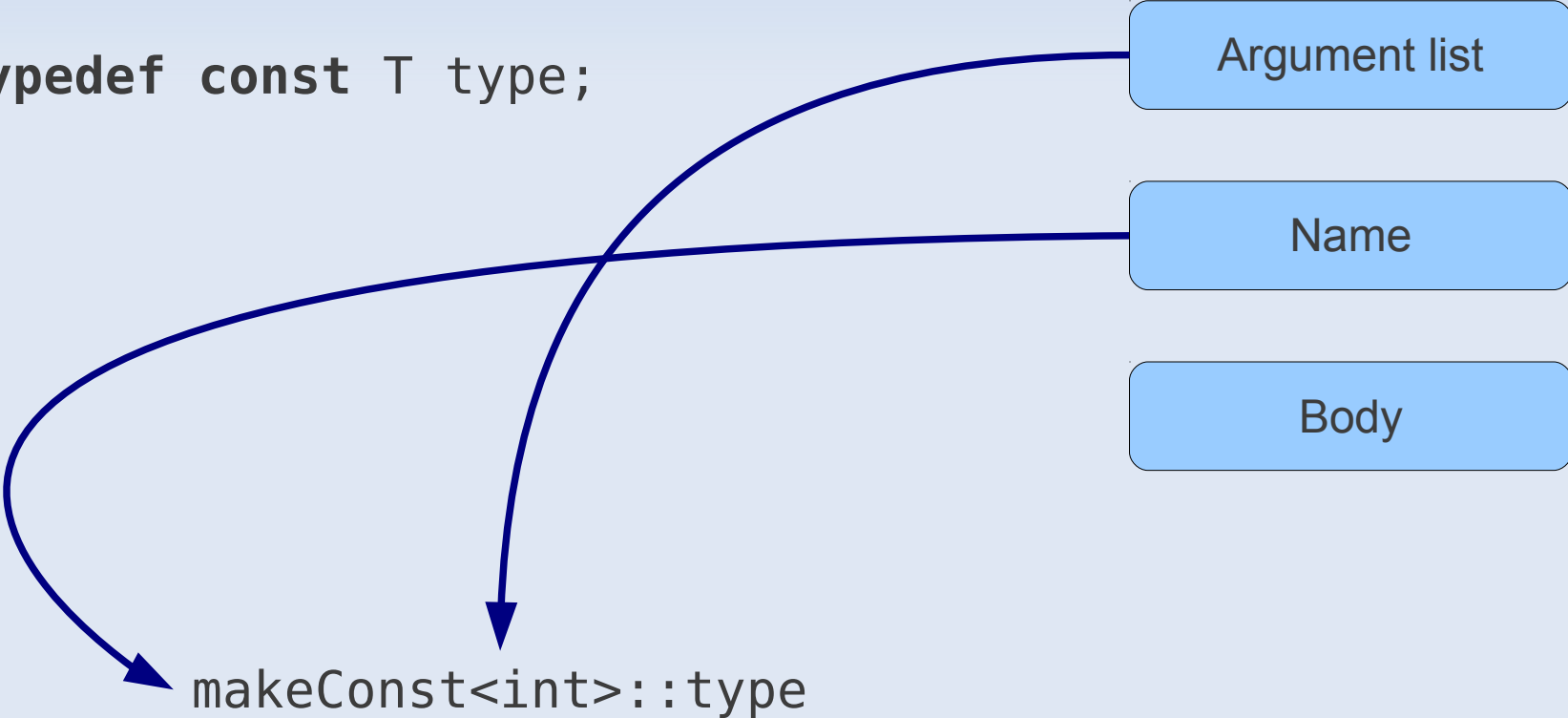
```
template <class T>  
struct makeConst  
{  
    typedef const T type;  
};
```

Argument list

Name

Body

makeConst<int>::type



Template metafunction class

```
struct makeConst  
{  
    template <class T>  
    struct apply  
    {  
        typedef const T type;  
    };  
};
```

Argument list

Name

Body

makeConst::apply<int>::type

Template metafunction class

```
struct makeConst
{
  template <class T>
  struct apply
  {
    typedef const T type;
  };
};
```

Argument list

Name

Body

makeConst::apply<int>::type

Example

```
template <class A, class B>  
struct min :  
  
{};
```

Example

```
template <class A, class B>  
struct min :  
    boost::mpl::if_<                >  
{};
```

Example

```
template <class A, class B>
struct min :
    boost::mpl::if_<less<A, B>,      >
{};
```

Example

```
template <class A, class B>
struct min :
    boost::mpl::if_<less<A, B>, A, B>
{};
```

Example

```
template <class A, class B>  
struct less;
```

```
template <class A, class B>  
struct min :  
    boost::mpl::if_<less<A, B>, A, B>  
{};
```

Example

```
template <class A, class B>  
struct less;
```

```
template <class A, class B>  
struct min :  
    boost::mpl::if_<less<A, B>, A, B>  
{};
```

```
template <class Re, class Im>  
struct complex;
```

Example

```
template <class A, class B>  
struct less;
```

```
template <class A, class B>  
struct min :  
    boost::mpl::if_<less<A, B>, A, B>  
{};
```

```
template <class Re, class Im>  
struct complex;
```

```
min<  
    complex<int_<19>, int_<83> >, // 19 + 83i  
    complex<int_<11>, int_<13> > // 11 + 13i  
>
```


Example

```
template <class A, class B>  
struct less;
```

```
template <class A, class B>  
struct min :  
    boost::mpl::if_<less<A, B>  
{};
```

```
template <class Re, class Im>  
struct complex;
```

```
min<  
    complex<int_<19>, int_<83> >, // 19 + 83i  
    complex<int_<11>, int_<13> > // 11 + 13i  
>
```

```
test.cpp: In instantiation of 'less<complex<mpl::int_<19>,  
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >':  
/usr/include/boost/mpl/if.hpp:67:11: instantiated from  
'boost::mpl::if_<less<complex<mpl::int_<19>, mpl::int_<83> >,  
complex<mpl::int_<11>, mpl::int_<13> > >, complex<mpl::int_<19>,  
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >'  
test.cpp:13:36: instantiated from 'min<complex<mpl::int_<19>,  
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >'  
test.cpp:21:68: instantiated from here  
test.cpp:10:44: error: 'value' is not a member of  
'complex<mpl::int_<11>, mpl::int_<13> >'  
test.cpp: In instantiation of 'less<complex<mpl::int_<19>,  
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >':  
/usr/include/boost/mpl/if.hpp:67:11: instantiated from  
'boost::mpl::if_<less<complex<mpl::int_<19>, mpl::int_<83> >,  
complex<mpl::int_<11>, mpl::int_<13> > >, complex<mpl::int_<19>,  
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >'  
test.cpp:13:36: instantiated from 'min<complex<mpl::int_<19>,  
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >'  
test.cpp:21:68: instantiated from here  
test.cpp:10:44: error: 'value' is not a member of  
'complex<mpl::int_<19>, mpl::int_<83> >'  
In file included from test.cpp:1:0:  
/usr/include/boost/mpl/if.hpp: In instantiation of  
'boost::mpl::if_<less<complex<mpl::int_<19>, mpl::int_<83> >,  
complex<mpl::int_<11>, mpl::int_<13> > >, complex<mpl::int_<19>,  
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >':  
test.cpp:13:36: instantiated from 'min<complex<mpl::int_<19>,  
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >'  
test.cpp:21:68: instantiated from here  
/usr/include/boost/mpl/if.hpp:67:11: error: 'value' is not a member of  
'less<complex<mpl::int_<19>, mpl::int_<83> >, complex<mpl::int_<11>,  
mpl::int_<13> > >'  
/usr/include/boost/mpl/if.hpp:70:41: error: 'value' is not a member of  
'less<complex<mpl::int_<19>, mpl::int_<83> >, complex<mpl::int_<11>,  
mpl::int_<13> > >'
```

Returning errors

```
template <class Reason>
struct exception
{
    typedef exception type;
};
```

Returning errors

```
template <class Reason>
struct exception
{
    typedef exception type;
};
```

```
template <class F>
struct debug_metafunction
{
    // tricks to display exception<Reason>
};
```

Returning errors

```
template <class Reason>
struct exception
{
    typedef exception type;
};
```

```
template <class F>
struct debug_metafunction
{
    // tricks to display exception<Reason>
};
```

```
struct values_can_not_be_compared;

// less<A, B> returns either
//     bool_<...>
//     exception<values_can_not_be_compared>
```

Example

```
template <class A, class B>
struct less :
    // returns either bool_<...>
    // or exception<values_can_not_be_compared>
    {};
```

```
template <class A, class B>
struct min :
    boost::mpl::if_<less<A, B>, A, B>
    {};
```

```
template <class Re, class Im>
struct complex;

min<
    complex<int_<19>, int_<83> >, // 19 + 83i
    complex<int_<11>, int_<13> > // 11 + 13i
>
```

Example

```
template <class A, class B>
struct less :
    // returns either bool_<..
    // or exception<values_can_
{};
```

```
template <class A, class B>
struct min :
    boost::mpl::if_<less<A, B>, A, B>
{};
```

```
In file included from test2.cpp:1:0:
/usr/include/boost/mpl/if.hpp: In instantiation of
'boost::mpl::if_<less<complex<mpl::int_<19>, mpl::int_<83> >,
complex<mpl::int_<11>, mpl::int_<13> > >, complex<mpl::int_<19>,
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >':
test2.cpp:21:36:   instantiated from 'min<complex<mpl::int_<19>,
mpl::int_<83> >, complex<mpl::int_<11>, mpl::int_<13> > >'
test2.cpp:29:68:   instantiated from here
/usr/include/boost/mpl/if.hpp:67:11: error: 'value' is not a member of
'less<complex<mpl::int_<19>, mpl::int_<83> >, complex<mpl::int_<11>,
mpl::int_<13> > >'
/usr/include/boost/mpl/if.hpp:70:41: error: 'value' is not a member of
'less<complex<mpl::int_<19>, mpl::int_<83> >, complex<mpl::int_<11>,
mpl::int_<13> > >'
```

```
template <class Re, class Im>
struct complex;
```

```
min<
    complex<int_<19>, int_<83> >, // 19 + 83i
    complex<int_<11>, int_<13> > // 11 + 13i
>
```

Error propagation

```
template <class A, class B>  
struct min :
```

```
    boost::mpl::if_<less<A, B>, A, B>
```

```
{};
```

```
template <class Re, class Im>  
struct complex;
```

```
min<
```

```
    complex<int_<19>, int_<83> >, // 19 + 83i
```

```
    complex<int_<11>, int_<13> > // 11 + 13i
```

```
>
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<

    boost::mpl::if_<less<A, B>, A, B>
>
{};
```

```
template <class Re, class Im>
struct complex;

min<
    complex<int_<19>, int_<83> >, // 19 + 83i
    complex<int_<11>, int_<13> > // 11 + 13i
>
```


Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        is_exception<less<A, B> > ,
        boost::mpl::if_<less<A, B>, A, B>
    >
{};
```

```
template <class Re, class Im>
struct complex;

min<
    complex<int_<19>, int_<83> >, // 19 + 83i
    complex<int_<11>, int_<13> > // 11 + 13i
>
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B> >::type,
        boost::mpl::if_<less<A, B>, A, B>
    >
    {};
```

```
template <class Re, class Im>
struct complex;

min<
    complex<int_<19>, int_<83> >, // 19 + 83i
    complex<int_<11>, int_<13> > // 11 + 13i
>
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B> >::type,
        less<A, B>,
        boost::mpl::if_<less<A, B>, A, B>
    >
    {};
```

```
template <class Re, class Im>
struct complex;

min<
    complex<int_<19>, int_<83> >, // 19 + 83i
    complex<int_<11>, int_<13> > // 11 + 13i
>
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B> >::type,
        less<A, B>,
        boost::mpl::if_<less<A, B>, A, B>
    >
    {};
```

exception<values_can_not_be_compared>

```
template <class Re, class Im>
struct complex;

min<
    complex<int_<19>, int_<83> >, // 19 + 83i
    complex<int_<11>, int_<13> > // 11 + 13i
>
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B> >::type,
        less<A, B>,
        boost::mpl::if_<less<A, B>, A, B>
    >
{};
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B> >::type,
        less<A, B>,
        boost::mpl::if_<less<A, B>, A, B>
    >
{};

struct min_impl
{
    template <class LessAB>
    struct apply : {}
};
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B> >::type,
        less<A, B>,
        boost::mpl::if_<less<A, B>, A, B>
    >
{};

struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl::if_<LessAB, A, B> {};
};
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B> >::type,
        less<A, B>,
        boost::mpl::if_<less<A, B>, A, B>
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl::if_<LessAB, A, B> {};
};
```


Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B> >::type,
        less<A, B>,
        boost::mpl::apply<min_impl<A, B>, less<A, B> >
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl::if_<LessAB, A, B> {};
};
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B>>::type,
        less<A, B>,
        boost::mpl::apply<min_impl<A, B>, less<A, B>>
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl::if_<LessAB, A, B> {};
};
```

Error propagation

```
template <class A, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception<less<A, B>>::type,
        less<A, B>,
        boost::mpl::apply<min_impl<A, B>, less<A, B>>
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl::if_<LessAB, A, B> {};
};
```

Error propagation

```
template <class X, class B>
struct min :
    boost::mpl::eval_if<
        typename is_exception< X >::type,
        X,
        boost::mpl::apply<min_impl<A, B>, X >
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl::if_<LessAB, A, B> {};
};
```

Error propagation

```
template <class X, class F>
struct min :
    boost::mpl::eval_if<
        typename is_exception< X >::type,
        X,
        boost::mpl::apply< F, X >
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl::if_<LessAB, A, B> {};
};
```

Error propagation

```
template <class X, class F>  
struct bind_exception :  
    boost::mpl::eval_if<  
        typename is_exception< X >::type,  
        X,  
        boost::mpl::apply< F , X >  
    >  
{};
```

```
template <class A, class B>  
struct min_impl  
{  
    template <class LessAB>  
    struct apply : boost::mpl::if_<LessAB, A, B> {};  
};
```

Error propagation

```
template <class X, class F>
struct bind_exception :
    boost::mpl::eval_if<
        typename is_exception< X >::type,
        X,
        boost::mpl::apply< F, X >
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl:
};
```

```
template <class A, class B>
struct min :
    bind_exception<
        less<A, B>,
        min_impl<A, B>
    >
{};
```

Error propagation

- A higher-order function:

×

→

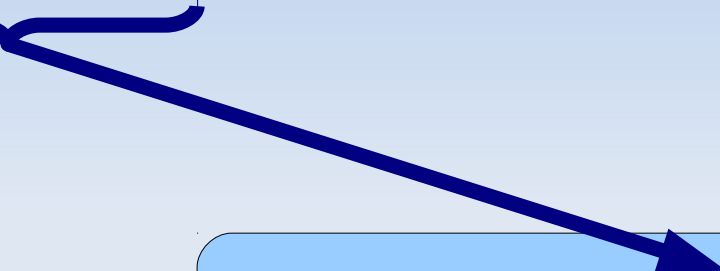
```
template <class X, class F>  
struct bind_exception;
```


Error propagation

- A higher-order function:

`error/result ×`

→

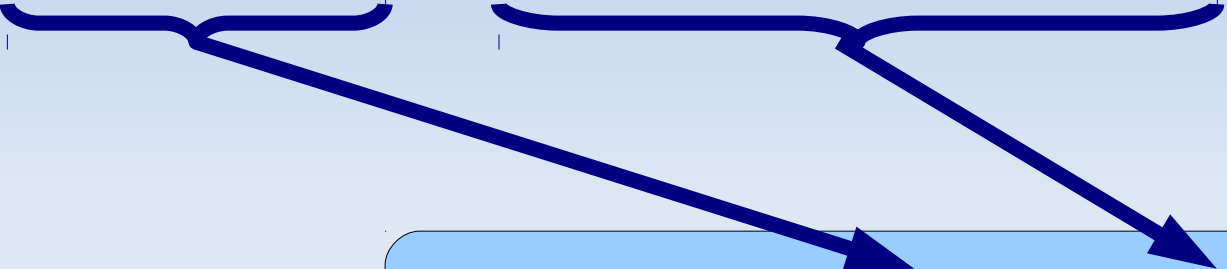


```
template <class X, class F>  
struct bind_exception;
```

Error propagation

- A higher-order function:

`error/result × (value → error/result) →`



```
template <class X, class F>  
struct bind_exception;
```

Error propagation

- A higher-order function:

`error/result × (value → error/result) → error/result`



```
template <class X, class F>  
struct bind_exception;
```

Error propagation

- Chaining functions

`error/result × (value → error/result) → error/result`

Error propagation

- Chaining functions

`error/result × (value → error/result) → error/result`

`error/result × (value → error/result) → error/result`



Error propagation

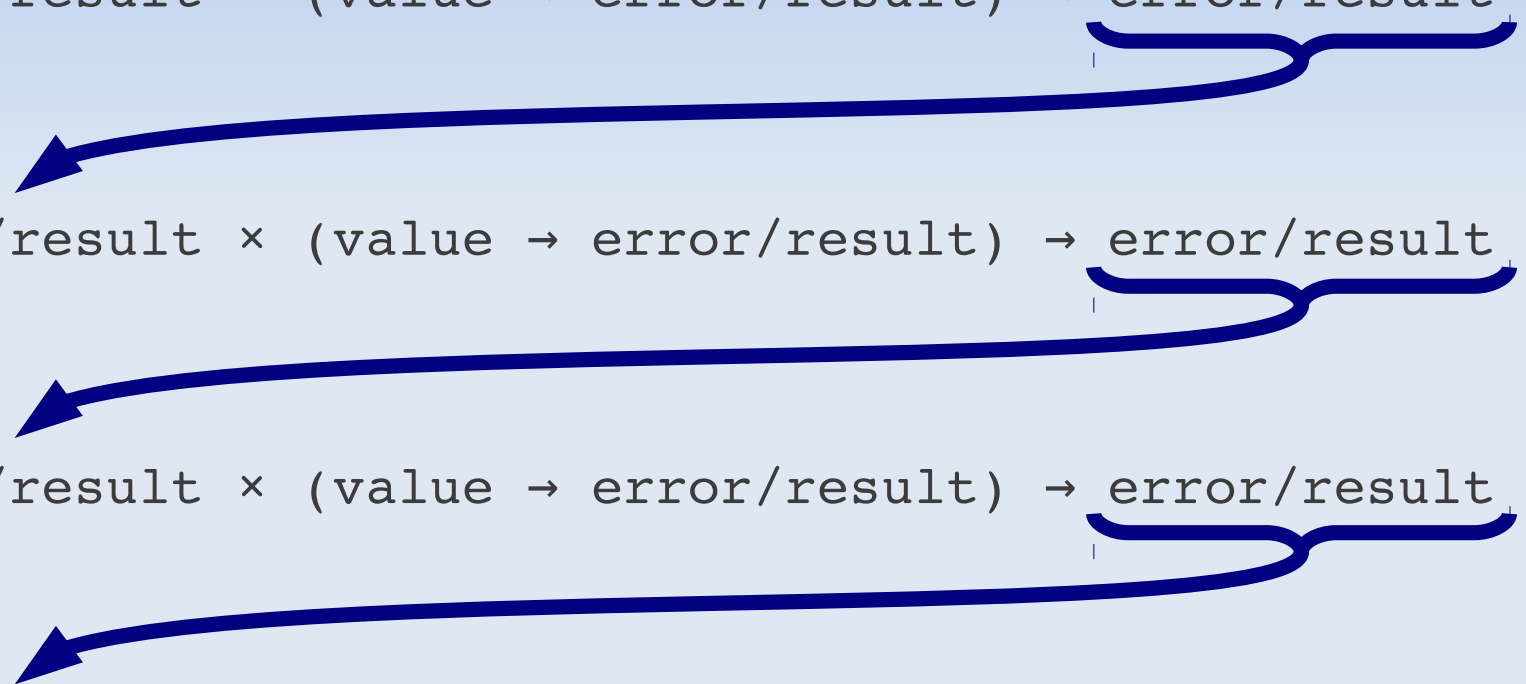
- Chaining functions

`error/result × (value → error/result) → error/result`

`error/result × (value → error/result) → error/result`

`error/result × (value → error/result) → error/result`

`error/result × (value → error/result) → error/result`



Error propagation

- Chaining functions

error/result × (value → error/result) → error/result

error/result × (value → error/result) → error/result

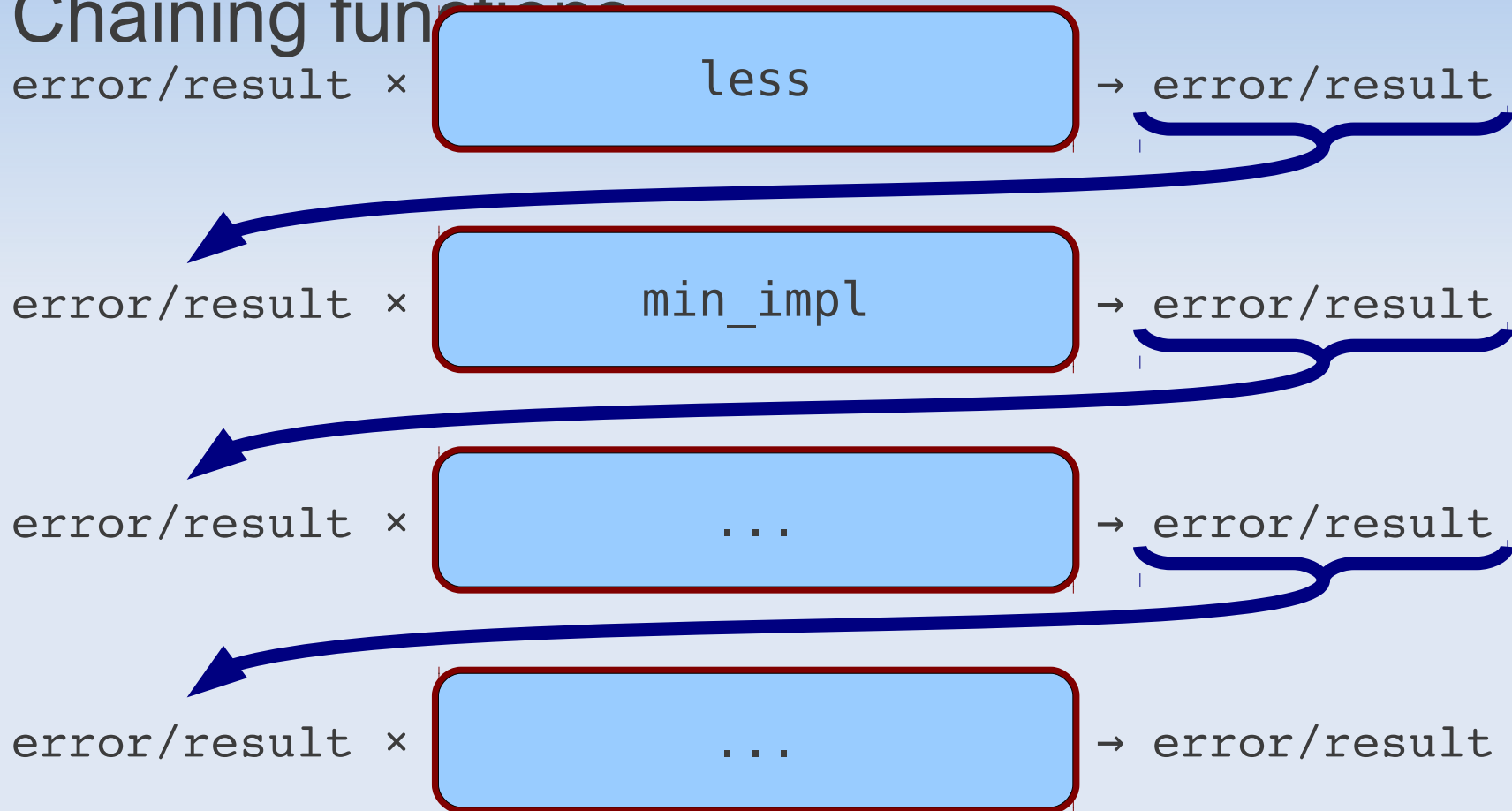
error/result × (value → error/result) → error/result

error/result × (value → error/result) → error/result

Business logic

Error propagation

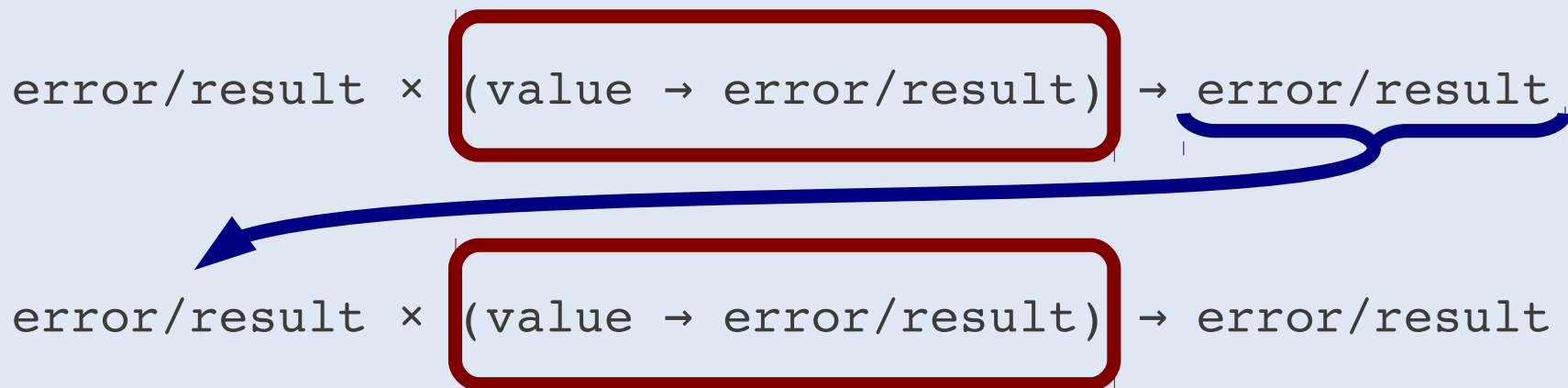
- Chaining functions



Business logic

Error propagation

- Starting the chain
value



Business logic

Error propagation

- Starting the chain

value

→

error/result

error/result × (value → error/result) → error/result

error/result × (value → error/result) → error/result

Business logic

Error propagation

- Starting the chain

value

→

error/result

error/result ×

(value → error/result)

→

error/result

error/result ×

(value → error/result)

→

error/result

Business logic

Error propagation

- We have two functions

Error propagation

- We have two functions

- `bind`

`error/result × (value → error/result) → error/result`

Error propagation

- We have two functions
- `bind`
`error/result × (value → error/result) → error/result`
- `return_`
`value → error/result`

Error propagation

- We have two functions

- `bind`



- `return`



Error propagation

- We have two functions

- `bind`



- `return`



- We have a set of values

T

Error propagation

- We have two functions

- `bind`



- `return`



- We have a set of values

T

- Further restrictions

Error propagation

- We have two functions

- `bind`



- `return`



- We have a set of values

T

- Further restrictions

Monads in metaprogramming

- A monad is:
 - A set of values

Monads in metaprogramming

- A monad is:
 - A set of values
 - Identified by a class (tag)

Monads in metaprogramming

- A monad is:
 - A set of values
 - Identified by a class (tag)
 - A `bind` metafunction

```
template <class Tag, class X, class F>  
struct bind;
```
 - A `return_` metafunction

```
template <class Tag, class X>  
struct return_;
```

Monads in metaprogramming

- A monad is:
 - A set of values
 - Identified by a class (tag)
 - A `bind` metafunction

```
template <class Tag, class X, class F>  
struct bind;
```
 - A `return_` metafunction

```
template <class Tag, class X>  
struct return_;
```
 - Further restrictions

Monads in Haskell

```
class Monad m where  
  (>>=)  :: m a -> (a -> m b) -> m b      -- bind  
  return :: a -> m a
```

The exception monad

- Set of values:
Every value in template metaprogramming
- Tag: **struct** exception_tag;
- bind:
template <class X, class F>
struct bind_exception;
- return_
template <class X>
struct identity;

min revisited

```
template <class A, class B>
struct min :
    bind<exception_tag,
        less<A, B>,
        min_impl<A, B>
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl::if_<LessAB, A, B> {};
};
```

min revisited

```
template <class A, class B>
struct min :
    bind<exception_tag,
        less<A, B>,
        min_impl<A, B>
    >
{};
```

```
template <class A, class B>
struct min_impl
{
    template <class LessAB>
    struct apply : boost::mpl
};
```

```
template <class A, class B>
struct min :
    DO<exception_tag>::apply<
        SET<x, less<A, B> >,
        boost::mpl::if_<x, A, B>
    >
{};
```

min revisited

```
template <class A, class B>
struct min :
    TRY<
        SET<x, less<A, B> >,
        boost::mpl::if_<x, A, B>
    >
    ::template catch_<some_tag, x>
    ::template apply< ... >
    ::template catch_<some_other_tag, x>
    ::template apply< ... >
{};
```

```
struct apply : boost::mpl::
};
```

```
class A, class B>
:
    <some_tag>::apply<
        SET<x, less<A, B> >,
        boost::mpl::if_<x, A, B>
    >
{};
```

Summary

- C++ template metaprogramming
- Current error handling approach
- New approach for error handling
- Monads in template metaprogramming

Q & A

<http://abel.web.elte.hu/mpllibs/>