Ranges for distributed and asynchronous systems

ACCU 2019, Bristol

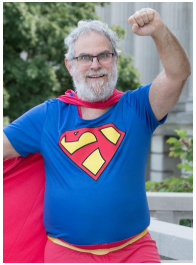
dr Ivan Čukić

ivan@cukic.co
http://cukic.co

About me

- Independent trainer / consultant
- KDE developer
- Author of the ”Functional Programming in C++” book
- University lecturer

Disclaimer



Make your code readable. Pretend the next person who looks at your code is a psychopath and they know where you live.

Philip Wadler

INTRODUCTION

Pointers

- Owned data (shared, unique)
- Non-owned data
- Reference to avoid copies
- **For iteration over arrays**

Iterators

```
std::copy_if(
    std::cbegin(items), std::cend(items),
    std::begin(output),
    matches);
```

- A pointer use-case abstraction
- A *simple* interface to elements in a collection
- Write once, run on any collection

Iterators

```
std::copy_if(
    std::cbegin(items), std::cend(items),
    std::begin(output),
    matches);
```

Input sequence

- A pointer use-case abstraction
- A *simple* interface to elements in a collection
- Write once, run on any collection

Iterators

```
std::copy_if(
    std::cbegin(items), std::cend(items),
    std::begin(output),
    matches);
```

Output sequence

- A pointer use-case abstraction
- A *simple* interface to elements in a collection
- Write once, run on any collection

Iterators

```
std::copy_if(
    std::cbegin(items), std::cend(items),
    std::back_inserter(output),
    matches);
```

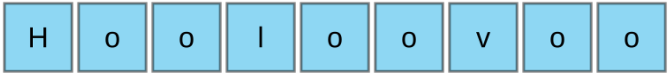
Appends values

Iterators

```
std::copy_if(
    line_iterator(std::cin), line_iterator(),
    std::ostream_iterator<std::string>(std::cout, '\n'),
    matches);
```

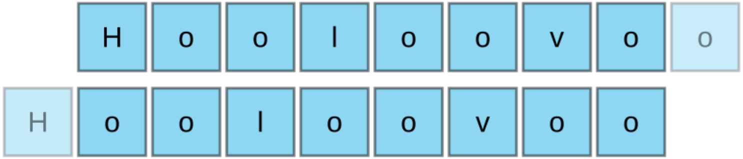
Composition

Task: Count repeated values



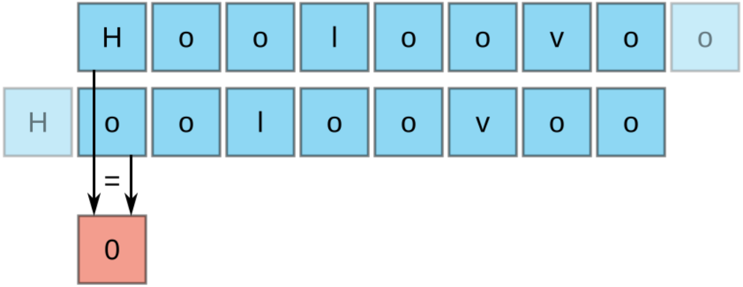
Composition

Task: Count repeated values



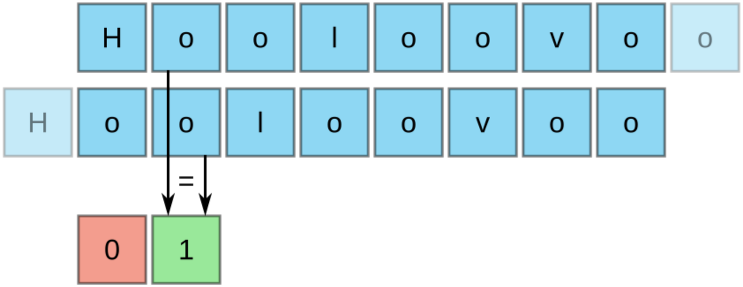
Composition

Task: Count repeated values



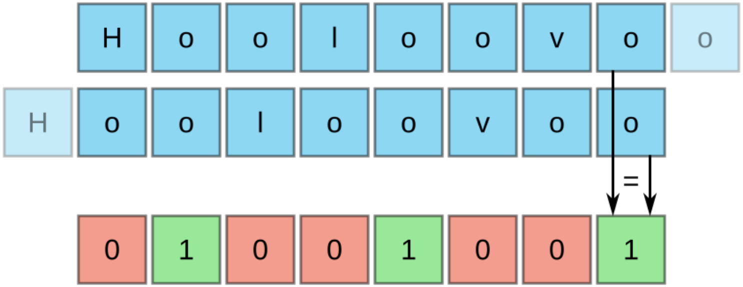
Composition

Task: Count repeated values



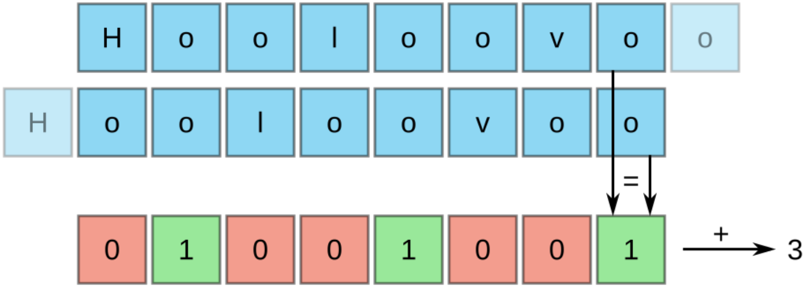
Composition

Task: Count repeated values



Composition

Task: Count repeated values



Composition

```
template <typename T>
int count_adj_equals(const T& xs)
{
    return std::inner_product(
        std::cbegin(xs), std::cend(xs) - 1,
        std::cbegin(xs) + 1,
        0,
        std::plus{},
        std::equal_to{});
}
```

To the penultimate el.

Composition

```
template <typename T>
int count_adj_equals(const T& xs)
{
    return std::inner_product(
        std::cbegin(xs), std::cend(xs) - 1,
        std::cbegin(xs) + 1,
        0,
        std::plus{},
        std::equal_to{});
}
```

Collection tail

Ranges

```
[ iterator, sentinel )
```

Iterator:

- `*i` – access the value
- `++i` – move to the next element

Sentinel:

- `i == s` – has iterator reached the end

Ranges

```
template <typename T>
int count_adj_equals(const T& xs)
{
    return accumulate(0,
               zip(xs, tail(xs)) | transform(equal_to{})
        );
}
```

* Not `std::equal_to`

Word frequency

1986: Donald Knuth was asked to implement a program for the ”Programming pearls” column in the Communications of ACM journal.

The task: Read a file of text, determine the n most frequently used words, and print out a sorted list of those words along with their frequencies.

Word frequency

1986: Donald Knuth was asked to implement a program for the ”Programming pearls” column in the Communications of ACM journal.

The task: Read a file of text, determine the n most frequently used words, and print out a sorted list of those words along with their frequencies.

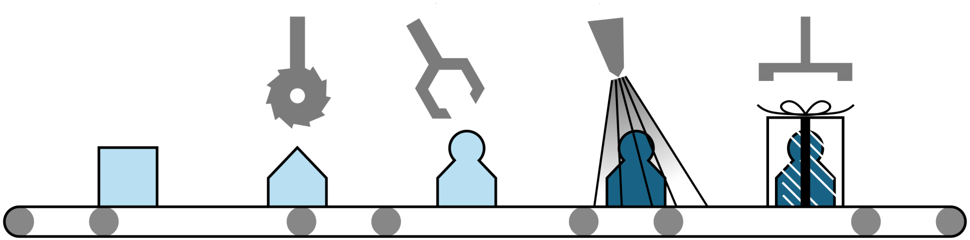
His solution written in Pascal was **10** pages long.

Word frequency

Response by Doug McIlroy was a 6-line shell script that did the same:

```
tr -cs A-Za-z '\n' |
    tr A-Z a-z |
    sort |
    uniq -c |
    sort -rn |
    sed ${1}q
```

Word frequency



Word frequency

```
std::string string_to_lower(const std::string &s) {
    return s | view::transform(tolower);
}

std::string string_only_alnum(const std::string &s) {
    return s | view::filter(isalnum);
}
```

Word frequency

```
const auto words =
      istream_range<std::string>(std::cin)
      | view::transform(string_to_lower)
      | view::transform(string_only_alnum)
      | view::remove_if(&std::string::empty)
      | to_vector | action::sort;
```

Word frequency

```
const auto results =
      words
      | view::group_by(std::equal_to())
      | view::transform([] (const auto &group) {
                const auto begin       = std::begin(group);
                const auto end         = std::end(group);
                const auto size        = distance(begin, end);
                const std::string word = *begin;

                return std::make_pair(size, word);
            })
      | to_vector | action::sort;
```

Word frequency

```
for (auto value: results | view::reverse
                         | view::take(n)
        ) {
    std::cout << value.first << " "
              << value.second << std::endl;
}
```

Ranges

```
[ iterator, sentinel )
```

Iterator:

- `*i` – access the value
- `++i` – move to the next element

Sentinel:

- `i == s` – has iterator reached the end

Ranges

```
[ iterator, sentinel )
```

Iterator:

- `*i` – access the value

BLOCKING!

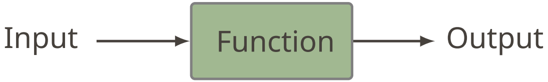
- `++i` – move to the next element

Sentinel:

- `i == s` – has iterator reached the end

PUSH

Push iterators



Push iterators

Each *push iterator* can:

- Accept values
- Emit values

No need for the accepted and emitted values to be 1-to-1.

Push iterators

Types of push iterators:

- Sources – push iterators that only emit values
- Sinks – push iterators that only accept values
- Transformations – push iterators that both accept and emit values

Continuation

```
template <typename Cont>
class continuator_base {
public:
    void init() { … }

    template <typename T>
    void emit(T&& value) const
    {
        std::invoke(m_continuation, FWD(value));
    }

    void notify_ended() const { … }

protected:
    Cont m_continuation;
};
```

Invoke

```
std::invoke(function, arg1, arg2, ...)
```

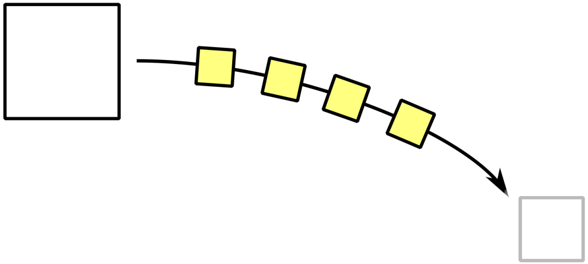
For most cases (functions, function objects, lambdas) equivalent to:

```
function(arg1, arg2, ...)
```

But it can also invoke class member functions:

```
arg1.function(arg2, ...)
```

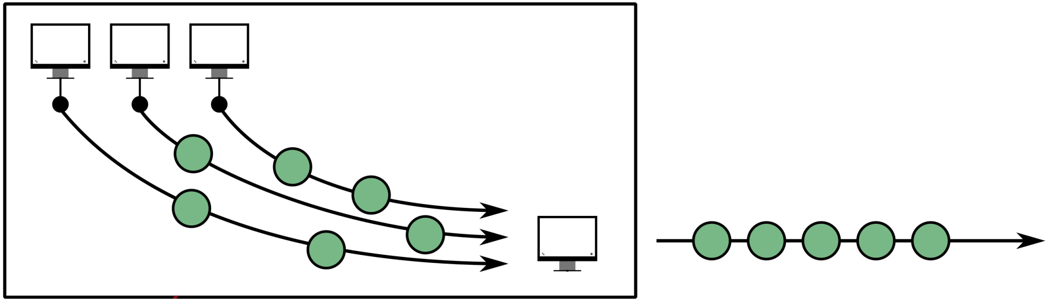
Source



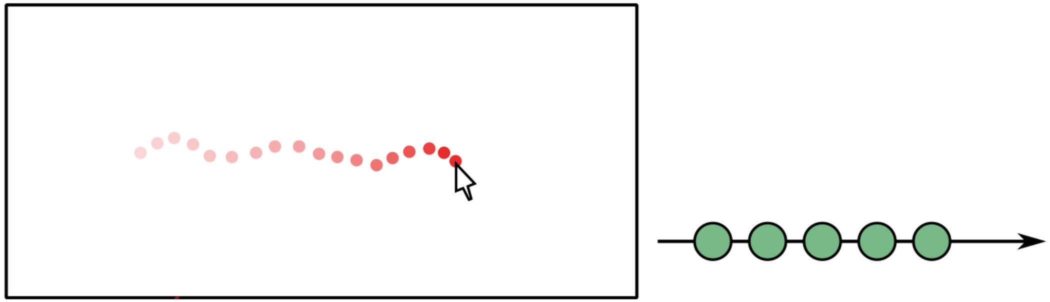
Pushing values

```
template <typename Cont>
class values_node: public continuator_base<Cont> {

    void init()
    {
        base::init();

        for (auto&& value: m_values) {
            base::emit(value);
        }

        m_values.clear();

        base::notify_ended();
    }
```

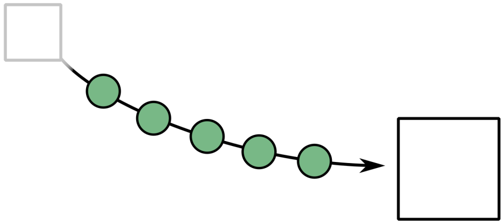
Creating a source



Creating a source



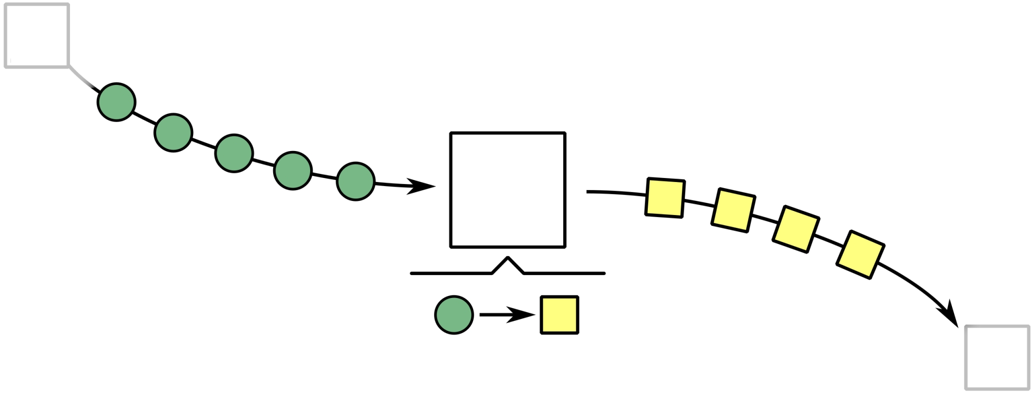
Creating a sink



Creating a sink

```
auto sink_to_cerr = sink([] (auto&& value) {
    std::cerr << FWD(value) << std::endl;
});

values{42, 6} | sink_to_cerr;

service(42042) | sink_to_cerr;

mouse() | sink_to_cerr;
```

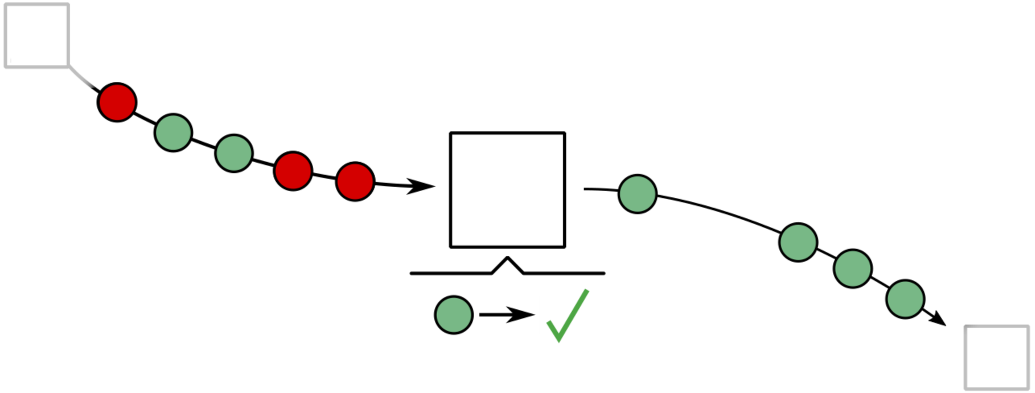
Creating a transformation



Creating a transformation

```
template <typename Cont>
class transform_node: public continuator_base<Cont> {
public:

    template <typename T>
    void operator() (T&& value) const
    {
        base::emit(std::invoke(m_transformation, FWD(value)));
    }

private:
    Traf m_transformation;
};
```

Creating a transformation



Filtering

```
template <typename Cont>
class filter_node: public continuator_base<Cont> {
public:
    template <typename T>
    void operator() (T&& value) const
    {
        if (std::invoke(m_predicate, value)) {
            base::emit(FWD(value)));
        }
    }

private:
    Predicate m_predicate;
};
```

PIPELINES

Pipelines

We want to create a simple web service.

- Line-based input
- Lines are JSON-encoded messages
- Each message is a bookmark – URL and the title
- And we will process the bookmarks

Pipelines

```
{ ”FirstURL”: ”https://isocpp.org/”, ”Text”: ”Standard C++” }
```

Boost.ASIO, 0mq or Qt

Pipelines

```
auto pipeline =
    service(42042)
        | transform(trim)
        | sink_to_cerr;
```

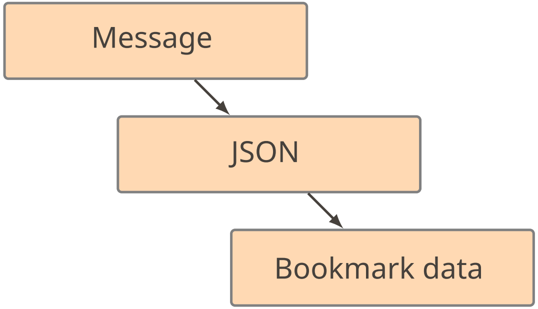
Pipelines

```
auto pipeline =
    service(42042)
        | transform(trim)
        | remove_if(&std::string::empty)
        | filter([] (const std::string& message) {
              return message[0] != '#';
          })
        | sink_to_cerr;
```

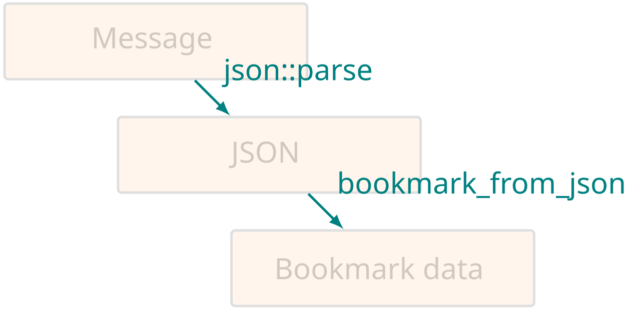
Pipelines

```
auto pipeline =
    values{ 42042, 42043, 42044 }
        | transform(make_service) | join()
        | transform(trim)
        | remove_if(&std::string::empty)
        | filter([] (const std::string& message) {
              return message[0] != '#';
          })
        | sink_to_cerr;
```

Pipelines



Pipelines



Pipelines

We have a stream of messages we need to parse.
Obvious choice is the `nlohmann/json` library.

```
{ "FirstURL": "https://isocpp.org/", "Text": "Standard C++" }
```

Exceptions?

Pipelines

```
template <typename F, typename Ret = …>
expected<Ret, std::exception_ptr> m_try(F f)
{
    try {
        return f();
    } catch (...) {
        return unexpected(std::current_exception());
    }
}
```

Pipelines

```
auto pipeline =
    ...

    | transform([] (std::string&& message) {
          return m_try([&] {
              return json::parse(message);
          });
      })

    | filter(&expected_json::is_valid)
    | transform(&expected_json::get)

    ...
```

We will get a stream of expected values

Pipelines

```
auto pipeline =
    ...

    | transform([] (std::string&& message) {
          return m_try([&] {
              return json::parse(message);
          });
      })

    | filter(&expected_json::is_valid)
    | transform(&expected_json::get)

    ...
```

And we retain only the valid ones

Pipelines

```
auto pipeline =
    …

    | transform(…)
    | filter(&expected_json::is_valid)
    | transform(&expected_json::get)

    | transform(json_to_bookmark)
    | filter(&expected_bookmark::is_valid)
    | transform(&expected_bookmark::get)

    …
```

Pipelines

If we have the need for error handling,
don’t work with streams of values,
but of streams of expected values.

Pipelines

```
auto pipeline =
    ...
    | transform([] (std::string&& message) {
          return m_try([&] {
              return json::parse(message);
          });
      })

    | transform([] (expected_json&& json) {
          return json.and_then(
              json_to_bookmark);
      })
    ...
```

Pipelines

- debouncing
- forking and merging
- value accumulation
- caching and buffering

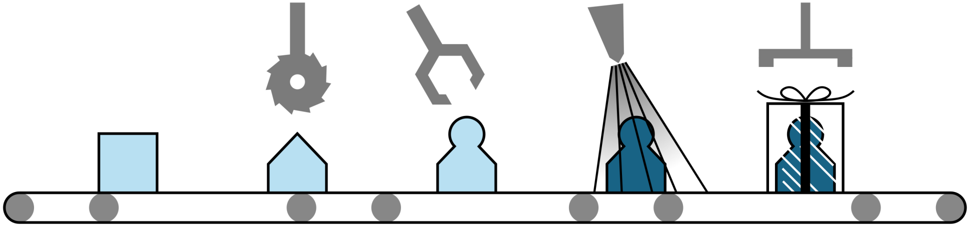
  ...

Pipelines

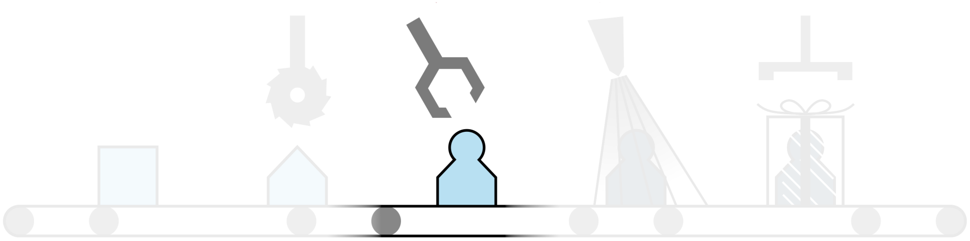
```
auto pipeline =
    ...

    | debounce(100ms)

    | tee(send_to_logger)

    | merge_with(control_events)

    ...
```

GOING POSTAL

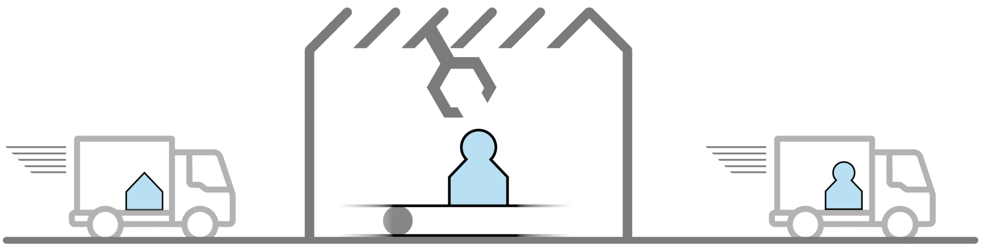
Going postal



Going postal



Going postal



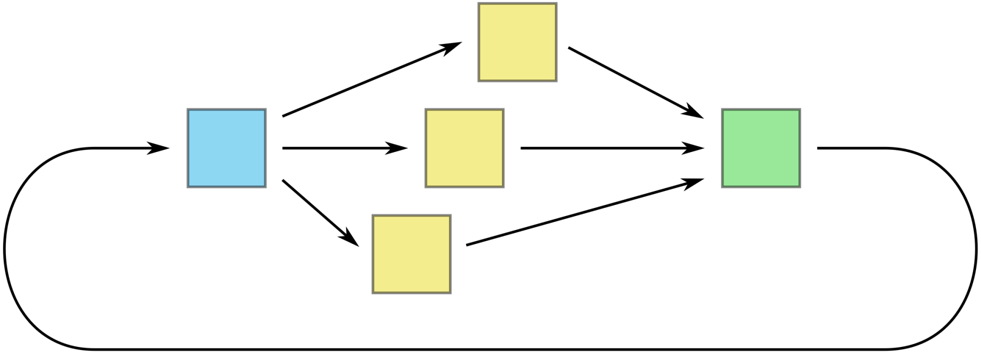
Going postal

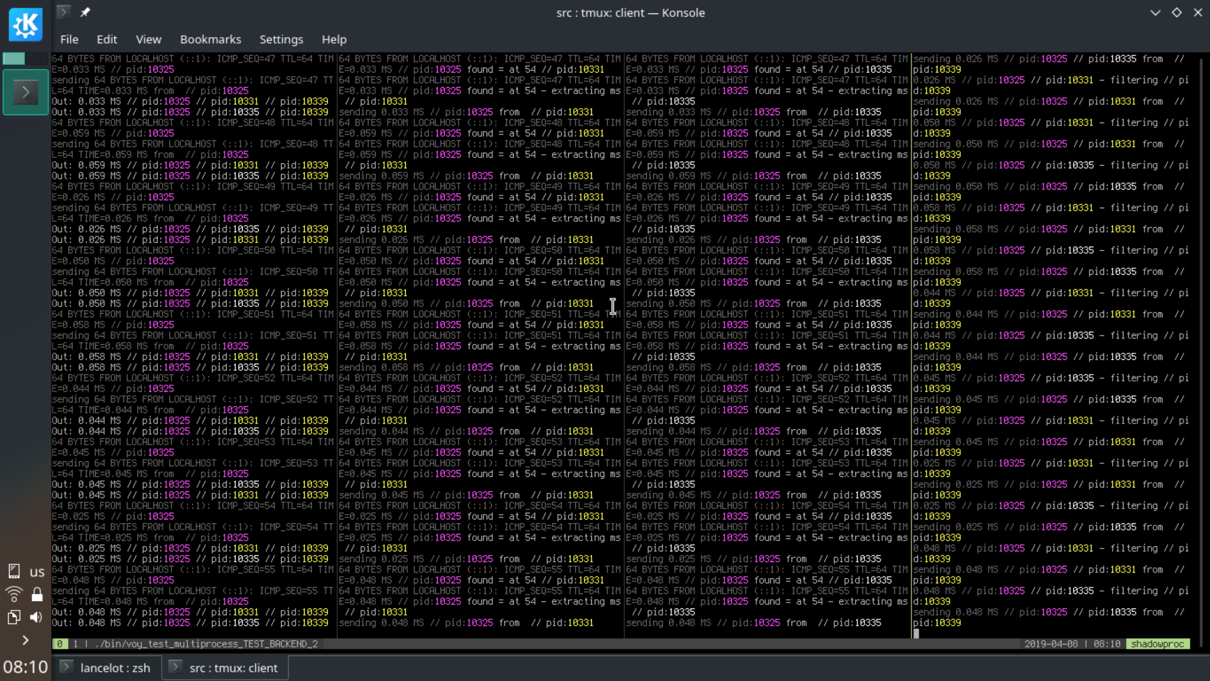
```
auto pipeline = system_cmd(”ping”s, ”localhost”s)
                | transform(string_to_upper)

                // Parse the ping output
                | transform([] (std::string&& value) {
                      const auto pos = value.find_last_of('=');
                      return std::make_pair(std::move(value), pos);
                  })

                // Extract the ping time from the output
                | transform([] (std::pair<std::string, size_t>&& pair) {
                      auto [ value, pos ] = pair;
                      return pos == std::string::npos ? std::move(value)
                                  : std::string(value.cbegin() + pos + 1, value.cend());
                  })

                // Remove slow pings
                | filter([] (const std::string& value) {
                      return value < ”0.145”s;
                  })

                | sink{cout};
```

Going postal

```
auto pipeline = system_cmd("ping"s, "localhost"s)
                | transform(string_to_upper)

                | voy_bridge(frontend_to_backend_1)

                | transform([] (std::string&& value) {
                      …
                  })

                | transform([] (std::pair<std::string, size_t>&& pair) {
                      …
                  })

                | voy_bridge(backend_1_to_backend_2)

                | filter([] (const std::string& value) {
                      return value < "0.145"s;
                  })

                | voy_bridge(backend_1_to_frontend)

                | sink{cout};
```

Going postal





```
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=47 TTL=64 TIM
E=0.033 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=47 TT
L=64 TIME=0.033 MS from  // pid:10325
Out: 0.033 MS // pid:10325 // pid:10331 // pid:10339
Out: 0.033 MS // pid:10325 // pid:10335 // pid:10339
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=48 TTL=64 TIM
E=0.059 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=48 TT
L=64 TIME=0.059 MS from  // pid:10325
Out: 0.059 MS // pid:10325 // pid:10331 // pid:10339
Out: 0.059 MS // pid:10325 // pid:10335 // pid:10339
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=49 TTL=64 TIM
E=0.026 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=49 TT
L=64 TIME=0.026 MS from  // pid:10325
Out: 0.026 MS // pid:10325 // pid:10335 // pid:10339
Out: 0.026 MS // pid:10325 // pid:10331 // pid:10339
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=50 TTL=64 TIM
E=0.050 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=50 TT
L=64 TIME=0.050 MS from  // pid:10325
Out: 0.050 MS // pid:10325 // pid:10331 // pid:10339
Out: 0.050 MS // pid:10325 // pid:10335 // pid:10339
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=51 TTL=64 TIM
E=0.058 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=51 TT
L=64 TIME=0.058 MS from  // pid:10325
Out: 0.058 MS // pid:10325 // pid:10331 // pid:10339
Out: 0.058 MS // pid:10325 // pid:10335 // pid:10339
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=52 TTL=64 TIM
E=0.044 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=52 TT
L=64 TIME=0.044 MS from  // pid:10325
Out: 0.044 MS // pid:10325 // pid:10331 // pid:10339
Out: 0.044 MS // pid:10325 // pid:10335 // pid:10339
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=53 TTL=64 TIM
E=0.045 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=53 TT
L=64 TIME=0.045 MS from  // pid:10325
Out: 0.045 MS // pid:10325 // pid:10335 // pid:10339
Out: 0.045 MS // pid:10325 // pid:10331 // pid:10339
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=54 TTL=64 TIM
E=0.025 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=54 TT
L=64 TIME=0.025 MS from  // pid:10325
Out: 0.025 MS // pid:10325 // pid:10331 // pid:10339
Out: 0.025 MS // pid:10325 // pid:10335 // pid:10339
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=55 TTL=64 TIM
E=0.048 MS // pid:10325
sending 64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=55 TT
L=64 TIME=0.048 MS from  // pid:10325
Out: 0.048 MS // pid:10325 // pid:10331 // pid:10339
Out: 0.048 MS // pid:10325 // pid:10335 // pid:10339
```

```
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=47 TTL=64 TIM
E=0.033 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=47 TTL=64 TIM
E=0.033 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.033 MS // pid:10325 from  // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=48 TTL=64 TIM
E=0.059 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=48 TTL=64 TIM
E=0.059 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.059 MS // pid:10325 from  // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=49 TTL=64 TIM
E=0.026 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=49 TTL=64 TIM
E=0.026 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.026 MS // pid:10325 from  // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=50 TTL=64 TIM
E=0.050 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=50 TTL=64 TIM
E=0.050 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.050 MS // pid:10325 from  // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=51 TTL=64 TIM
E=0.058 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=51 TTL=64 TIM
E=0.058 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.058 MS // pid:10325 from  // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=52 TTL=64 TIM
E=0.044 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=52 TTL=64 TIM
E=0.044 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.044 MS // pid:10325 from  // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=53 TTL=64 TIM
E=0.045 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=53 TTL=64 TIM
E=0.045 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.045 MS // pid:10325 from  // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=54 TTL=64 TIM
E=0.025 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=54 TTL=64 TIM
E=0.025 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.025 MS // pid:10325 from  // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=55 TTL=64 TIM
E=0.048 MS // pid:10325 found = at 54 // pid:10331
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=55 TTL=64 TIM
E=0.048 MS // pid:10325 found = at 54 - extracting ms
 // pid:10331
sending 0.048 MS // pid:10325 from  // pid:10331
```

```
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=47 TTL=64 TIM
E=0.033 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=47 TTL=64 TIM
E=0.033 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.033 MS // pid:10325 from  // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=48 TTL=64 TIM
E=0.059 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=48 TTL=64 TIM
E=0.059 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.059 MS // pid:10325 from  // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=49 TTL=64 TIM
E=0.026 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=49 TTL=64 TIM
E=0.026 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.026 MS // pid:10325 from  // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=50 TTL=64 TIM
E=0.050 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=50 TTL=64 TIM
E=0.050 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.050 MS // pid:10325 from  // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=51 TTL=64 TIM
E=0.058 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=51 TTL=64 TIM
E=0.058 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.058 MS // pid:10325 from  // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=52 TTL=64 TIM
E=0.044 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=52 TTL=64 TIM
E=0.044 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.044 MS // pid:10325 from  // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=53 TTL=64 TIM
E=0.045 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=53 TTL=64 TIM
E=0.045 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.045 MS // pid:10325 from  // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=54 TTL=64 TIM
E=0.025 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=54 TTL=64 TIM
E=0.025 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.025 MS // pid:10325 from  // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=55 TTL=64 TIM
E=0.048 MS // pid:10325 found = at 54 // pid:10335
64 BYTES FROM LOCALHOST (::1): ICMP_SEQ=55 TTL=64 TIM
E=0.048 MS // pid:10325 found = at 54 - extracting ms
 // pid:10335
sending 0.048 MS // pid:10325 from  // pid:10335
```

```
sending 0.026 MS // pid:10325 // pid:10335 from  //
pid:10339
0.026 MS // pid:10325 // pid:10331 - filtering // pi
d:10339
sending 0.026 MS // pid:10325 // pid:10331 from  //
pid:10339
0.050 MS // pid:10325 // pid:10331 - filtering // pi
d:10339
sending 0.050 MS // pid:10325 // pid:10331 from  //
pid:10339
0.050 MS // pid:10325 // pid:10335 - filtering // pi
d:10339
sending 0.050 MS // pid:10325 // pid:10335 from  //
pid:10339
0.058 MS // pid:10325 // pid:10331 - filtering // pi
d:10339
sending 0.058 MS // pid:10325 // pid:10331 from  //
pid:10339
0.058 MS // pid:10325 // pid:10335 - filtering // pi
d:10339
sending 0.058 MS // pid:10325 // pid:10335 from  //
pid:10339
0.044 MS // pid:10325 // pid:10331 - filtering // pi
d:10339
sending 0.044 MS // pid:10325 // pid:10331 from  //
pid:10339
0.044 MS // pid:10325 // pid:10335 - filtering // pi
d:10339
sending 0.044 MS // pid:10325 // pid:10335 from  //
pid:10339
0.045 MS // pid:10325 // pid:10335 - filtering // pi
d:10339
sending 0.045 MS // pid:10325 // pid:10335 from  //
pid:10339
0.045 MS // pid:10325 // pid:10331 - filtering // pi
d:10339
sending 0.045 MS // pid:10325 // pid:10331 from  //
pid:10339
0.025 MS // pid:10325 // pid:10331 - filtering // pi
d:10339
sending 0.025 MS // pid:10325 // pid:10331 from  //
pid:10339
0.025 MS // pid:10325 // pid:10335 - filtering // pi
d:10339
sending 0.025 MS // pid:10325 // pid:10335 from  //
pid:10339
0.048 MS // pid:10325 // pid:10331 - filtering // pi
d:10339
sending 0.048 MS // pid:10325 // pid:10331 from  //
pid:10339
0.048 MS // pid:10325 // pid:10335 - filtering // pi
d:10339
sending 0.048 MS // pid:10325 // pid:10335 from  //
pid:10339
```

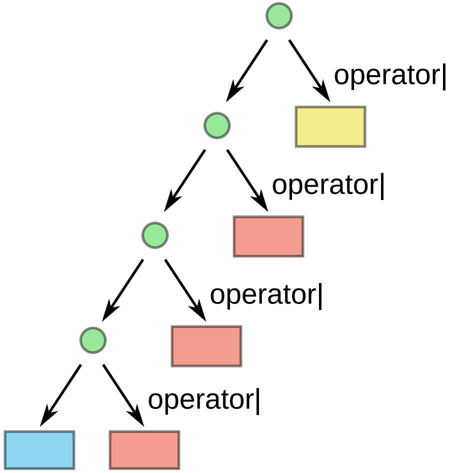
```
0 1 | ./bin/voy_test_multiprocess_TEST_BACKEND_2                    2019-04-08 | 08:10 shadowproc
```

IMPLEMENTATION

Syntax

```
service(42042)
        | transform(trim)
        | remove_if(&std::string::empty)
        | filter([] (const std::string& message) {
              return message[0] != '#';
          })
        | sink_to_cerr;
```

Syntax



Syntax

```
template <typename... Nodes>
class expression {
    template <typename Continuation>
    auto operator| (Continuation&& cont) &&
    {
        ...
    }
};
```

Syntax

`std::function`: type erasure is cool but slow.

Use a right-associative operator »= to appease Haskell gods?

Syntax

`std::function`: type erasure is cool but slow.

Expression templates to the rescue!

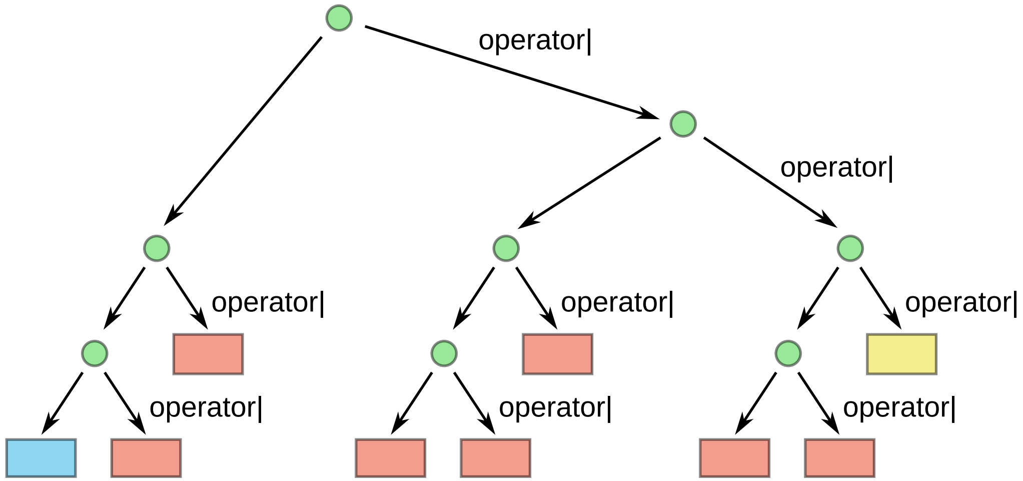
Syntax

```
template <typename... Nodes>
class expression {
    template <typename Continuation>
    auto operator| (Continuation&& cont) &&
    {
        return expression(
            std::tuple_cat(
                std::move(m_nodes),
                std::make_tuple(FWD(cont))));
    }

    std::tuple<Nodes...> m_nodes;
};
```

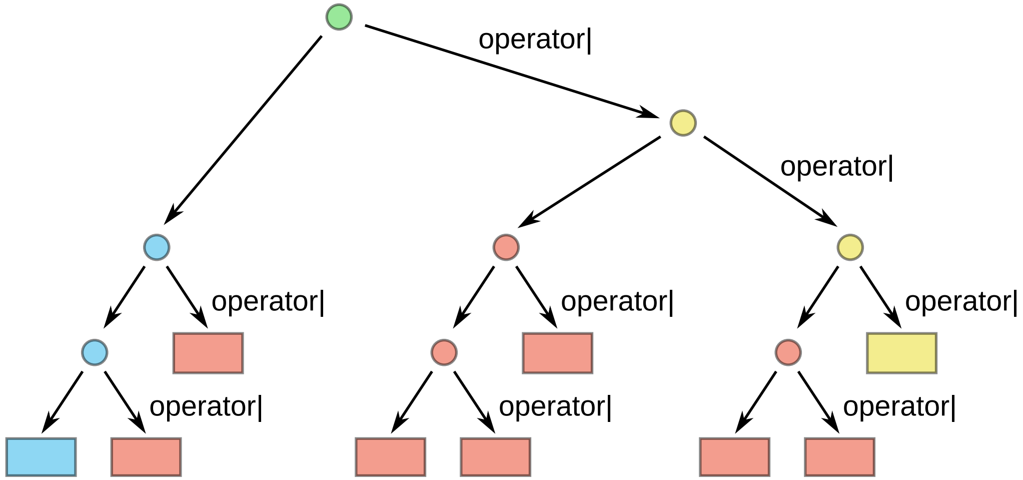
Syntax

```
auto user_names = users | transform(&user_t::name);
auto ignore_empty = transform(trim)
                  | remove_if(&std::string::empty);

user_names | ignore_empty | transform(string_to_upper);
```

Syntax


operator|
operator|
operator|
operator|
operator|
operator|
operator|
operator|

Syntax


operator|
operator|
operator|
operator|
operator|
operator|
operator|
operator|

Syntax

- Different meanings of `operator|`
- Wildly different types of operands (no inheritance tree)
- Arbitrary complex AST

Universal expression

```
template <typename Left, typename Right>
struct expression {
    Left left;
    Right right;
};
```

```
<node> ::= <producer> | <consumer> | <trafo> | <expression>
<expression> ::= <node> <|> <node>
```

Meta information

Adding meta-information to classes:

```
struct producer_node_tag {};
struct consumer_node_tag {};
struct transformation_node_tag {};

class filter_node {
public:
    using node_type_tag =
        transformation_node_tag;
};
```

Meta information

```
template <typename Node>
using node_category =
    typename remove_cvref_t<Node>::node_type_tag;
```

Universal expression

```
template <typename Tag, typename Left, typename Right>
struct expression {
    using node_type_tag = Tag;

    Left left;
    Right right;
};
```

Meta information

```
template < typename Node
         , typename Category =
               std::detected_t<node_category, Node>
constexpr bool is_node()
{
    if constexpr (!is_detected_v<node_category, Node>) {
        return false;

    } else if constexpr (
            std::is_same_v<complete_pipeline_tag, Category>) {
        return false;

    } else {
        return true;
    }
}
```

Restricting the pipe

```
template < typename Left
         , typename Right
         , REQUIRE(is_node<Left>() && is_node<Right>())
         >
auto operator| (Left&& left, Right&& right)
{
    ...








}
```

Restricting the pipe

```
template < typename Left
         , typename Right
         , REQUIRE(is_node<Left>() && is_node<Right>())
         >
auto operator| (Left&& left, Right&& right)
{
    if constexpr (!is_producer<Left> && !is_consumer<Right>) {
        return expression<transformation_node_tag, Left, Right>{
            FWD(left), FWD(right)
        };
    }


    ...
}
```

Restricting the pipe

```
template < typename Left
         , typename Right
         , REQUIRE(is_node<Left>() && is_node<Right>())
         >
auto operator| (Left&& left, Right&& right)
{
    … else
    if constexpr (is_producer<Left> && !is_consumer<Right>) {
        return expression<producer_node_tag, Left, Right>{
            FWD(left), FWD(right)
        };
    }

    …
}
```

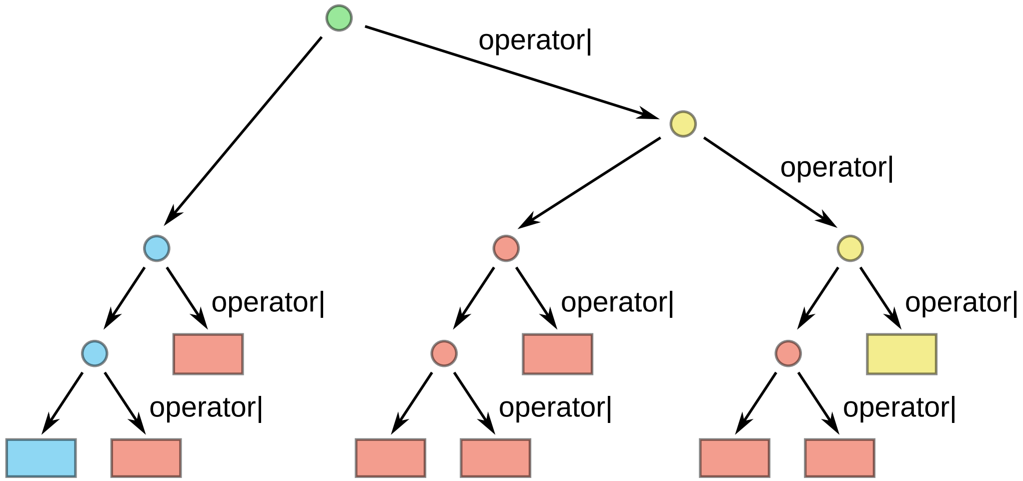
Restricting the pipe

```
template < typename Left
         , typename Right
         , REQUIRE(is_node<Left>() && is_node<Right>())
         >
auto operator| (Left&& left, Right&& right)
{

    … else
    if constexpr (!is_producer<Left> && is_consumer<Right>) {
        return expression<consumer_node_tag, Left, Right>{
            FWD(left), FWD(right)
        };
    }
    …
}
```

Restricting the pipe

```
template < typename Left
         , typename Right
         , REQUIRE(is_node<Left>() && is_node<Right>())
         >
auto operator| (Left&& left, Right&& right)
{


    … else
    if constexpr (is_producer<Left> && is_consumer<Right>) {
        return expression<complete_pipeline_tag, Left, Right>{
            FWD(left), FWD(right)
        };
    }
}
```

Evaluation


operator|
operator|
operator|
operator|
operator|
operator|
operator|
operator|

AST transformation

1. Collect nodes from the left sub-tree
2. Collect nodes from the right sub-tree
3. Merge the results

AST transformation

```
template <typename Expr>
auto collect_nodes(Expr&& expr)
{
    auto collect_sub_nodes = [] (auto&& sub) {
        if constexpr (is_expression<decltype(sub)>) {
            return collect_nodes(std::move(sub));
        } else {
            return std::make_tuple(std::move(sub));
        }
    };

    return std::tuple_cat(
        collect_sub_nodes(std::move(expr.left)),
        collect_sub_nodes(std::move(expr.right)));
}
```

Evaluation

Two choices:

- Connect left-to-right
- Connect right-to-left

LTR

Pros:

- Easier
- Easy to pass `value_type` around

Cons:

- Type erasure

RTL

Pros:

- No need for type erasure

Cons:

- No way to pass `value_type`:

```
service(42042) | debounce<std::string>(200ms) | ...
```

Both!

Feed forward and backward connect.

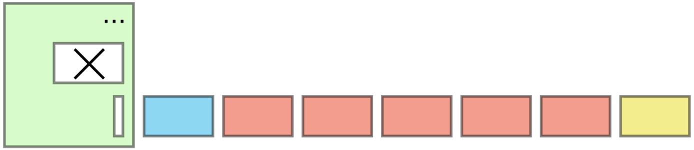
Context propagation

```
struct transform_t {

    template <typename In>
    using value_type_for_input_t = …

};
```

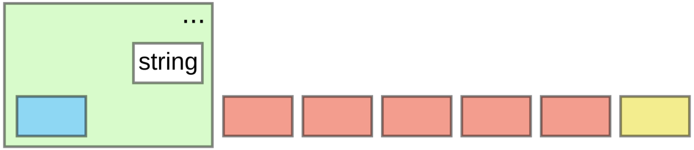
Context propagation

```
using new_value_type =
    typename Data::template value_type_for_input_t<ValueType>;
```

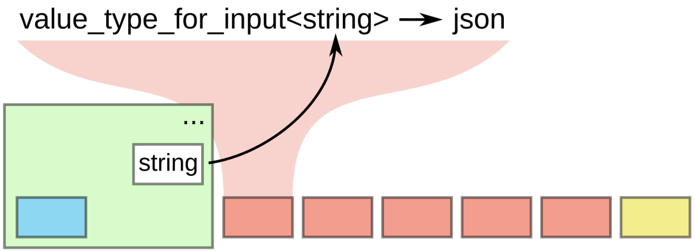
Context propagation



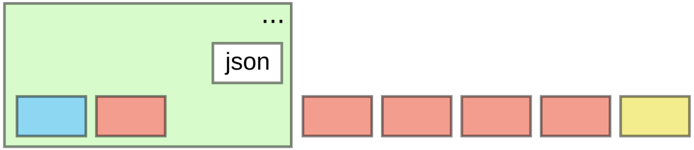
Context propagation



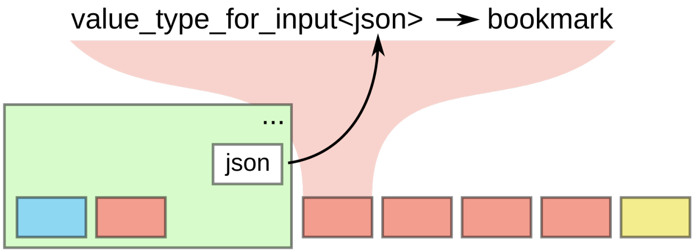
Context propagation



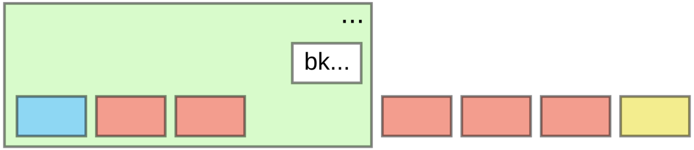
Context propagation



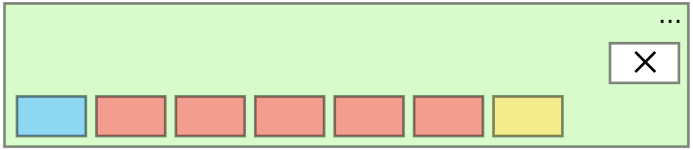
Context propagation



Context propagation



Context propagation



Context propagation

Connection

Now we have a list of enriched nodes,
we can connect them right-to-left.

Evaluation

```
template <typename... Nodes>
auto evaluate_nodes(Nodes&&... nodes)
{
    return (... % nodes);
}
```

Connection

```
template <typename Node, typename Connected>
auto operator% (Node&& new_node, Connected&& connected)
{
    return FWD(new_node).with_continuation(FWD(connected));
}
```

Summary

Abstractions:

- over collections
- over values
- over connections

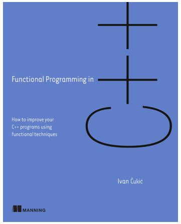
Answers? Questions! Questions? Answers!

Reaching me

Web: https://**cukic.co**
Mail: **ivan@cukic.co**
Twitter: **@ivan_cukic**

Kudos (in chronological order)

Friends at **KDE**
Saša Malkov and **Zoltan Porkolab**
Сергей Платонов


Functional Programming in
How to improve your C++ programs using functional techniques
Ivan Čukić
MANNING

cukic.co/to/fp-in-cpp
Functional Programming in C++

Discount code
ctwaccu19
(40% of all Manning books)