

CPPCON 2018

# STATE MACHINES BATTLEFIELD

## NAIVE VS STL VS BOOST

Kris Jusiak, Quantlab Financial

[KRIS@JUSIAK.NET](mailto:KRIS@JUSIAK.NET) | [@KRISJUSIAK](https://twitter.com/KRISJUSIAK) | [LINKEDIN.COM/IN/KRIS-JUSIAK](https://www.linkedin.com/in/kris-jusiak)

# PROBLEM / MOTIVATION

---

---

# PROBLEM / MOTIVATION

Feature: Connection # BDD style

---

# PROBLEM / MOTIVATION

Feature: Connection # BDD style

---

Scenario: Establish connection

---

# PROBLEM / MOTIVATION

Feature: Connection # BDD style

---

Scenario: Establish connection

Given I don't have a connection

# PROBLEM / MOTIVATION

Feature: Connection # BDD style

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

---

# PROBLEM / MOTIVATION

Feature: Connection # BDD style

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

Then I should try to establish the connection

---

# PROBLEM / MOTIVATION

Feature: Connection # BDD style

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

Then I should try to establish the connection

When I receive an established acknowledgement

---



# PROBLEM / MOTIVATION

Feature: Connection # BDD style

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

Then I should try to establish the connection

When I receive an established acknowledgement

Then I should have been connected

---

# PROBLEM / MOTIVATION

Feature: Connection # BDD style

---

Scenario: Establish connection

Given I don't have a connection

When I receive a request to connect

Then I should try to establish the connection

When I receive an established acknowledgement

Then I should have been connected

---

Scenario: Disconnect

...

# PROBLEM / MOTIVATION

*What's the*

- 
- 
- 

*way to satisfy/implement Connection  
requirements?*

# PROBLEM / MOTIVATION

*What's the*

- most readable
- 
- 

*way to satisfy/implement Connection requirements?*

# PROBLEM / MOTIVATION

*What's the*

- most readable
- most maintainable
- 

*way to satisfy/implement Connection requirements?*

# PROBLEM / MOTIVATION

*What's the*

- most readable
- most maintainable
- most efficient

*way to satisfy/implement Connection requirements?*

# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5



# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

*State Machine - Connection*

[HTTPS://WWW.OMG.ORG/SPEC/UML/2.5.1/PDF](https://www.omg.org/spec/UML/2.5.1/PDF)



# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

## *State Machine - Connection*

```
Feature: Connection
  Scenario: Connect
    ...

  Scenario: Disconnect
    ...

  Scenario: ...
    ...
```

[HTTPS://WWW.OMG.ORG/SPEC/UML/2.5.1/PDF](https://www.omg.org/spec/UML/2.5.1/PDF)

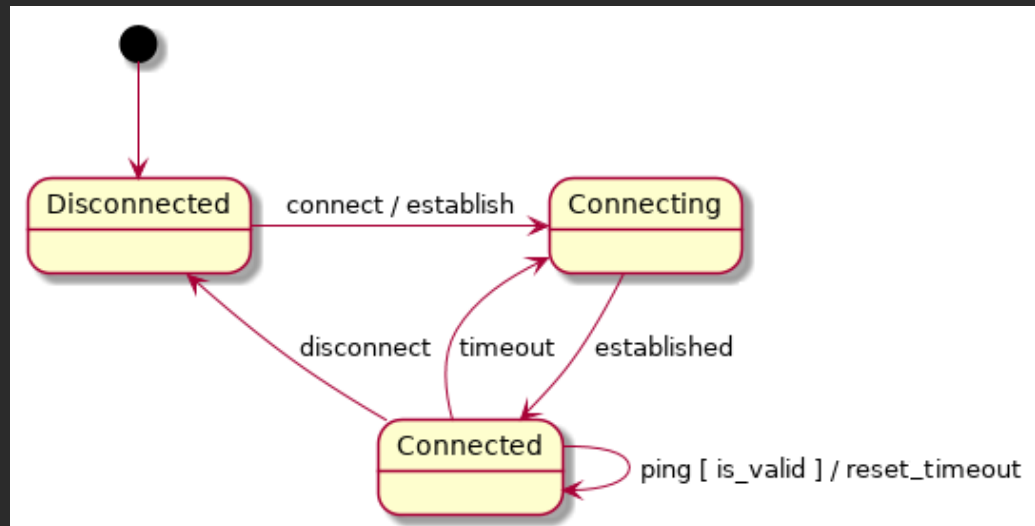
# STATE MACHINE - UNIFIED MODELING LANGUAGE - 2.5

## *State Machine - Connection*

```
Feature: Connection
  Scenario: Connect
    ...

  Scenario: Disconnect
    ...

  Scenario: ...
    ...
```



[HTTPS://WWW.OMG.ORG/SPEC/UML/2.5.1/PDF](https://www.omg.org/spec/UML/2.5.1/PDF)

# SOLUTIONS

# SOLUTIONS

Naive	STL	Boost
If/Else (C++98)	std::variant (C++17)	Boost.Statechart (C++98)
Switch/Enum (C++98)	Coroutines (C++20)	Boost.MSM (C++98)
Inheritance / State pattern (C++98)		[Boost].SML (C++14)

# COMMON - IMPLEMENTATION

# COMMON - IMPLEMENTATION

## EVENTS

```
struct connect{};  
struct established{};  
struct ping{};  
struct disconnect{};  
struct timeout{};
```

# COMMON - IMPLEMENTATION

## EVENTS

```
struct connect{};  
struct established{};  
struct ping{};  
struct disconnect{};  
struct timeout{};
```

## GUARDS

```
constexpr auto is_valid      = [](auto const& event) { return true; };
```

# COMMON - IMPLEMENTATION

## EVENTS

```
struct connect{};  
struct established{};  
struct ping{};  
struct disconnect{};  
struct timeout{};
```

## GUARDS

```
constexpr auto is_valid      = [] (auto const& event) { return true; };
```

## ACTIONS

```
constexpr auto establish    = [] { std::puts("establish"); };  
constexpr auto close       = [] { std::puts("close"); };  
constexpr auto reset_timeout = [] { std::puts("reset_timeout"); };
```



# NAIVE

*C++98 features*

# NAIVE - IF/ELSE - IMPLEMENTATION

## NAIVE - IF/ELSE - IMPLEMENTATION

```
class Connection {  
    // Implicit states using booleans  
    bool disconnected = true, connected = false, connecting = false;
```

# NAIVE - IF/ELSE - IMPLEMENTATION

```
class Connection {  
    // Implicit states using booleans  
    bool disconnected = true, connected = false, connecting = false;  
  
    constexpr void process_event(connect const&) {  
        if (disconnected) {  
            establish();  
            connected = disconnected = false; // Just in case reset it all!  
            connecting = true; // Set the new state  
        }  
    }  
}
```

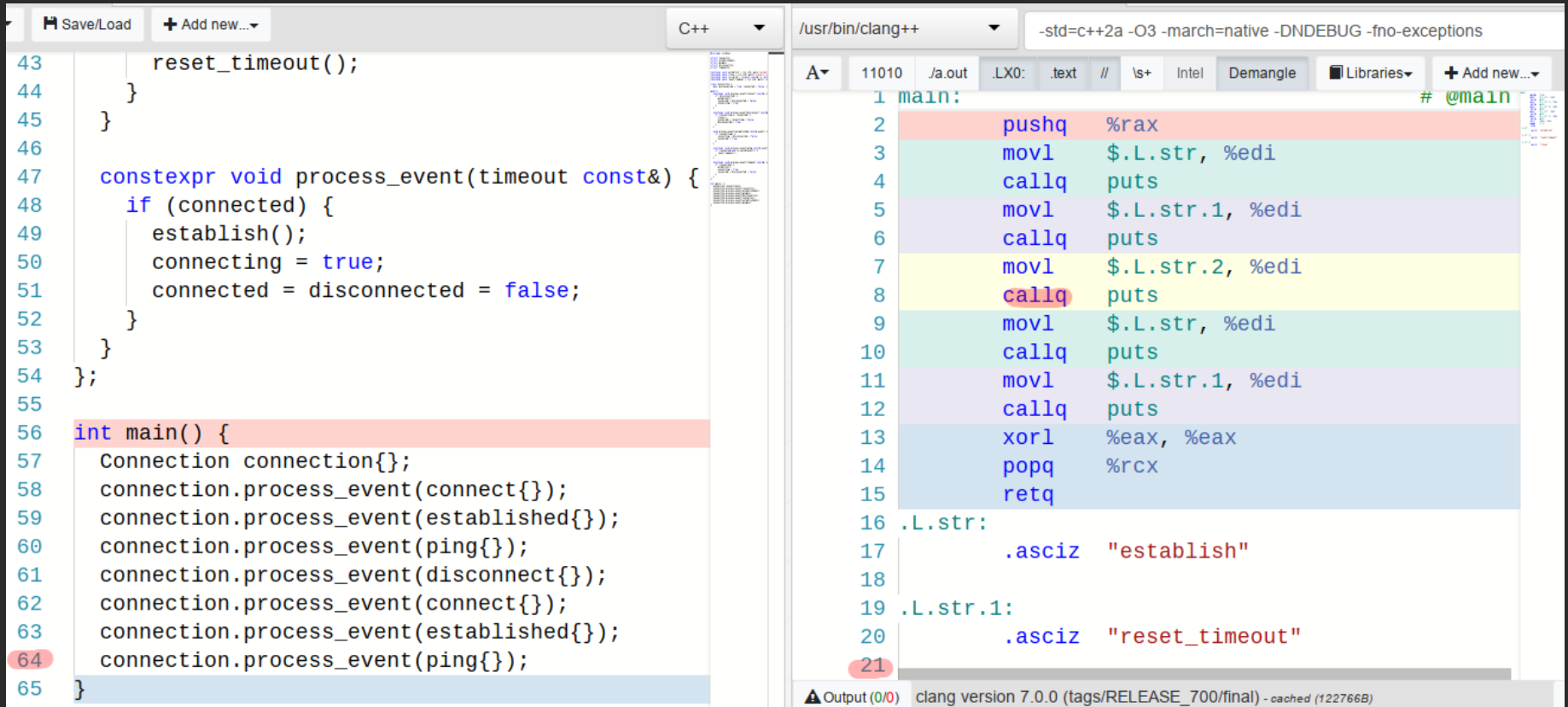
# NAIVE - IF/ELSE - IMPLEMENTATION

```
class Connection {  
    // Implicit states using booleans  
    bool disconnected = true, connected = false, connecting = false;
```

```
constexpr void process_event(connect const&) {  
    if (disconnected) {  
        establish();  
        connected = disconnected = false; // Just in case reset it all!  
        connecting = true; // Set the new state  
    }  
}
```

```
constexpr void process_event(ping const& event) {  
    if (connected and is_valid(event)) {  
        reset_timeout();  
    } // Stay in the current state  
}
```

# NAIVE - IF/ELSE - FULL EXAMPLE



The image shows a C++ IDE with two panes. The left pane displays C++ source code, and the right pane displays the corresponding assembly code generated by clang++.

**C++ Source Code (Left Pane):**

```
43     reset_timeout();
44 }
45 }
46
47 constexpr void process_event(timeout const&) {
48     if (connected) {
49         establish();
50         connecting = true;
51         connected = disconnected = false;
52     }
53 }
54 };
55
56 int main() {
57     Connection connection{};
58     connection.process_event(connect{});
59     connection.process_event(established{});
60     connection.process_event(ping{});
61     connection.process_event(disconnect{});
62     connection.process_event(connect{});
63     connection.process_event(established{});
64     connection.process_event(ping{});
65 }
```

**Assembly Code (Right Pane):**

```
1 main:                                     # @main
2     pushq   %rax
3     movl    $.L.str, %edi
4     callq   puts
5     movl    $.L.str.1, %edi
6     callq   puts
7     movl    $.L.str.2, %edi
8     callq   puts
9     movl    $.L.str, %edi
10    callq   puts
11    movl    $.L.str.1, %edi
12    callq   puts
13    xorl    %eax, %eax
14    popq    %rcx
15    retq
16 .L.str:
17     .asciz  "establish"
18
19 .L.str.1:
20     .asciz  "reset_timeout"
21
```

The assembly code shows the translation of the C++ code. It includes a `main` function that calls `puts` for each event type. The string literals are stored in memory and pointed to by the `edi` register. The `retq` instruction at the end of the function indicates the return of the program.

[HTTPS://GODBOLT.ORG/Z/APHWNC](https://godbolt.org/z/APHWNC)

# NAIVE - IF/ELSE - SUMMARY

## NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)



## NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)
- (+) No heap usage

## NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)
- (+) No heap usage
- (~) Small-ish memory footprint
  - `sizeof(Connection) == 3b`

## NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)
- (+) No heap usage
- (~) Small-ish memory footprint
  - `sizeof(Connection) == 3b`
- (-) Hard to reuse

# NAIVE - IF/ELSE - SUMMARY

- (+) Inlined (gcc/clang)
- (+) No heap usage
- (~) Small-ish memory footprint
  - `sizeof(Connection) == 3b`
- (-) Hard to reuse

```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] == $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```



# NAIVE - SWITCH/ENUM - IMPLEMENTATION

# NAIVE - SWITCH/ENUM - IMPLEMENTATION

```
class Connection {  
    // Only one state can be active  
    enum class State : char { DISCONNECTED,  
                               CONNECTING,  
                               CONNECTED } state = DISCONNECTED;
```

# NAIVE - SWITCH/ENUM - IMPLEMENTATION

```
class Connection {  
    // Only one state can be active  
    enum class State : char { DISCONNECTED,  
                               CONNECTING,  
                               CONNECTED } state = DISCONNECTED;
```

```
constexpr void process_event(connect const&) {  
    switch (state) { // Handle current state  
        default: break;  
        case State::DISCONNECTED:  
            establish(); state = State::CONNECTING; break;  
            // Set the new state  
    }  
}
```

# NAIVE - SWITCH/ENUM - IMPLEMENTATION

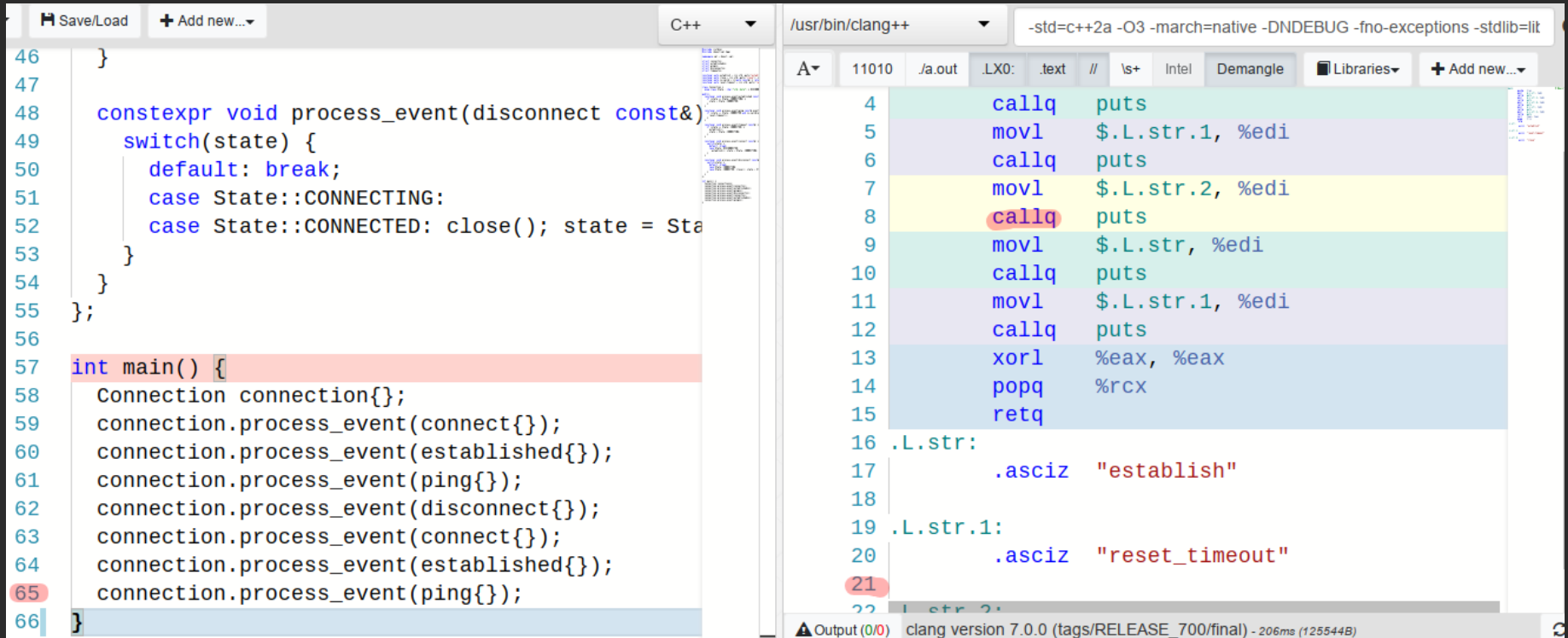
```
class Connection {  
    // Only one state can be active  
    enum class State : char { DISCONNECTED,  
                               CONNECTING,  
                               CONNECTED } state = DISCONNECTED;
```

```
constexpr void process_event(connect const&) {  
    switch (state) { // Handle current state  
        default: break;  
        case State::DISCONNECTED:  
            establish(); state = State::CONNECTING; break;  
            // Set the new state  
    }  
}
```

```
constexpr void process_event(ping const& event) {  
    switch (state) {  
        default: break;  
        case State::CONNECTED:  
            if (is_valid(event)) { reset_timeout(); }  
            break; // Stay in the current state  
    }  
}
```



# NAIVE - SWITCH/ENUM - FULL EXAMPLE



The screenshot shows a C++ IDE with two panels. The left panel displays the source code for a program using a naive switch statement. The right panel shows the corresponding assembly code generated by clang++.

**Source Code (Left Panel):**

```
46 }
47
48 constexpr void process_event(disconnect const&)
49 {
50     switch(state) {
51         default: break;
52         case State::CONNECTING:
53         case State::CONNECTED: close(); state = Sta
54     }
55 };
56
57 int main() {
58     Connection connection{};
59     connection.process_event(connect{});
60     connection.process_event(established{});
61     connection.process_event(ping{});
62     connection.process_event(disconnect{});
63     connection.process_event(connect{});
64     connection.process_event(established{});
65     connection.process_event(ping{});
66 }
```

**Assembly Code (Right Panel):**

```
A 11010 /a.out LX0: .text // \s+ Intel Demangle Libraries+ Add new...
4      callq puts
5      movl $.L.str.1, %edi
6      callq puts
7      movl $.L.str.2, %edi
8      callq puts
9      movl $.L.str, %edi
10     callq puts
11     movl $.L.str.1, %edi
12     callq puts
13     xorl %eax, %eax
14     popq %rcx
15     retq
16 .L.str:
17     .asciz "establish"
18
19 .L.str.1:
20     .asciz "reset_timeout"
21
22 .L.str.2:
```

The assembly code shows a sequence of calls to the `puts` function, each preceded by a `movl` instruction that loads a string address into the `%edi` register. The strings are "establish" and "reset\_timeout".

[HTTPS://GODBOLT.ORG/Z/NM\\_-OY](https://godbolt.org/z/NM_-OY)

# NAIVE - SWITCH/ENUM - SUMMARY

## NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)

## NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)
- (+) Small memory footprint
  - `sizeof(Connection) == 1b`

## NAIVE - SWITCH/ENUM - SUMMARY

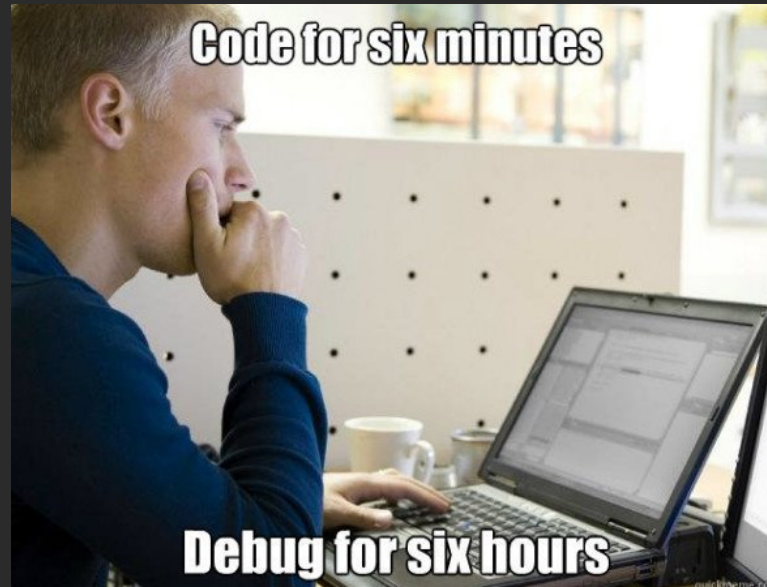
- (+) Inlined (gcc/clang)
- (+) Small memory footprint
  - `sizeof(Connection) == 1b`
- (+) No heap usage

## NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)
- (+) Small memory footprint
  - `sizeof(Connection) == 1b`
- (+) No heap usage
- (-) Hard to reuse

# NAIVE - SWITCH/ENUM - SUMMARY

- (+) Inlined (gcc/clang)
- (+) Small memory footprint
  - `sizeof(Connection) == 1b`
- (+) No heap usage
- (-) Hard to reuse



# INHERITANCE / STATE PATTERN - IMPLEMENTATION



## INHERITANCE / STATE PATTERN - IMPLEMENTATION

```
struct State {  
    virtual ~State() noexcept = default;  
    virtual void process_event(connect const&) = 0;  
    virtual void process_event(ping const&) = 0;  
    virtual void process_event(established const&) = 0;  
    virtual void process_event(timeout const&) = 0;  
    virtual void process_event(disconnect const&) = 0;  
};
```

# INHERITANCE / STATE PATTERN - IMPLEMENTATION

# INHERITANCE / STATE PATTERN - IMPLEMENTATION

```
struct Disconnected : State {  
    Connection& connection;  
  
    void process_event(connect const&) override final {  
        establish();  
        connection.change_state<Connecting>();  
    }  
};
```

# INHERITANCE / STATE PATTERN - IMPLEMENTATION

```
struct Disconnected : State {
    Connection& connection;

    void process_event(connect const&) override final {
        establish();
        connection.change_state<Connecting>();
    }
};
```

```
struct Connected : State {
    Connection& connection;

    void process_event(ping const& event) override final {
        if (is_valid(event)) {
            reset_timeout();
        }
    }

    // ...
};
```

# INHERITANCE / STATE PATTERN - FULL EXAMPLE

The image shows a C++ IDE with two main panels. The left panel displays the source code for a State Pattern implementation, and the right panel shows the assembly output generated by the compiler.

**Source Code (Left Panel):**

```
84     connection.change_state<Connecting>();
85 }
86
87 void process_event(disconnect const&) override
88     close();
89     connection.change_state<Disconnected>();
90 }
91
92 private:
93     Connection& connection;
94 };
95
96 int main() {
97     Connection connection{};
98     connection.init_state<Disconnected>();
99     connection.process_event(connect{});
100    connection.process_event(established{});
101    connection.process_event(ping{});
102    connection.process_event(disconnect{});
103    connection.process_event(connect{});
104    connection.process_event(established{});
105    connection.process_event(ping{});
106 }
```

**Assembly Output (Right Panel):**

The right panel shows the assembly output for the compiled code. It includes the clang version 7.0.0 output and the g++ (GCC) 8.2.0 output.

**clang version 7.0.0 (tags/RELEASE\_700/final) - 29ms (801852B)**

```
222     .quad    typeinfo for State
223
224 .L.str.1:
225     .asciz   "reset_timeout"
226
227 .L.str.2:
228     .asciz   "close"
```

**g++ (GCC) 8.2.0 - 53ms (567715B)**

```
115     ret
116 main:
117     subq    $40, %rsp
118     movl    $vtable for Disconnected+16, %edi
119     movl    $16, %edi
120     leaq    24(%rsp), %rax
121     vmovq   %rcx, %xmm0
122     movq    $0, 24(%rsp)
```

[HTTPS://GODBOLT.ORG/Z/DUI-AR](https://godbolt.org/z/DUI-AR)

# INHERITANCE / STATE PATTERN - SUMMARY

## INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)

## INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)
- (~) High-ish memory footprint



## INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)
- (~) High-ish memory footprint
- (-) Heap usage / dynamic allocations

## INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)
- (~) High-ish memory footprint
- (-) Heap usage / dynamic allocations
- (-) Not inlined/devirtualized (even with final)

# INHERITANCE / STATE PATTERN - SUMMARY

- (+) Easy to extend/reuse (object oriented)
- (~) High-ish memory footprint
- (-) Heap usage / dynamic allocations
- (-) Not inlined/devirtualized (even with final)



# STL

*C++17 / C++20 Standard Template  
Library*

# STD::VARIANT - IMPLEMENTATION

# STD::VARIANT - IMPLEMENTATION

```
class Connection {  
    struct Disconnected { }; // May have additional data  
    struct Connecting { };  
    struct Connected { };  
  
    // Only one active state  
    std::variant<Disconnected, Connecting, Connected> state  
        = Disconnected{};
```

# STD::VARIANT - IMPLEMENTATION

```
class Connection {
    struct Disconnected { }; // May have additional data
    struct Connecting { };
    struct Connected { };

    // Only one active state
    std::variant<Disconnected, Connecting, Connected> state
        = Disconnected{};
```

```
constexpr void process_event(connect const&) {
    std::visit(overload{ // Choose one of the following...
        [&](Disconnected) { establish(); state = Connecting{}; },
        [](auto)           { } // No changes...
    }, state);
}
```

# STD::VARIANT - IMPLEMENTATION

```
class Connection {
    struct Disconnected { }; // May have additional data
    struct Connecting { };
    struct Connected { };

    // Only one active state
    std::variant<Disconnected, Connecting, Connected> state
        = Disconnected{};
```

```
constexpr void process_event(connect const&) {
    std::visit(overload{ // Choose one of the following...
        [&](Disconnected) { establish(); state = Connecting{}; },
        [](auto)           { } // No changes...
    }, state);
}
```

```
void process_event(ping const& event) {
    if (std::get_if<Connected>(&state) and is_valid(event)) {
        reset_timeout();
    } // Stay in the current state
}
```



# STD::VARIANT - FULL EXAMPLE

The image shows a C++ IDE with two main panels. The left panel displays the source code, and the right panel shows the compiled assembly code.

**Source Code (Left Panel):**

```
49     if (std::get_if<Connected>(&state) and is_val
50         reset_timeout();
51     }
52 }
53
54 void process_event(timeout const&) {
55     if (std::get_if<Connected>(&state)) {
56         establish();
57         state = Connecting{};
58     }
59 }
60 };
61
62 int main() {
63     Connection connection{};
64     connection.process_event(connect{});
65     connection.process_event(established{});
66     connection.process_event(ping{});
67     connection.process_event(disconnect{});
68     connection.process_event(connect{});
69     connection.process_event(established{});
70     connection.process_event(ping{});
71 }
```

**Assembly Code (Right Panel):**

The right panel shows the assembly code generated by Clang 7.0.0. The top section shows the `main` function, and the bottom section shows the `main.cold.48` section.

**main:**

```
1  main:                                     # @main
2      pushq   %rax
3      movl    $.L.str, %edi
4      callq   puts
5      movl    $.L.str.1, %edi
6      callq   puts
7      movl    $.L.str.2, %edi
8      callq   puts
```

**main.cold.48:**

```
134     movl    $.LC3, %edi
135     call     puts
136     jmp      .L54
137 main.cold.48:
138 .L53:
139     call     abort
140 std::__detail::__variant::__gen_vtable<void, ove
141     quad     std::__detail::__variant::__gen
```

[HTTPS://GODBOLT.ORG/Z/OY2FBL](https://godbolt.org/z/OY2FBL)

# STD::VARIANT - SUMMARY

## STD::VARIANT - SUMMARY

- (+) Small/efficient memory footprint

- ```
Disconnect { };
Connecting { connection_id id{}; };
Connected  { connection_id id{};
            time last_ping{}; };
```

## STD::VARIANT - SUMMARY

- (+) Small/efficient memory footprint

- ```
Disconnect { };
Connecting { connection_id id{}; };
Connected  { connection_id id{};
            time last_ping{}; };
```

- (+) Integrates well with std::expected/static exceptions

- ```
return Error{"timeout"};
```

## STD::VARIANT - SUMMARY

- (+) Small/efficient memory footprint

- ```
Disconnect { };  
Connecting { connection_id id{}; };  
Connected  { connection_id id{};  
            time last_ping{}; };
```

- (+) Integrates well with std::expected/static exceptions

- ```
return Error{"timeout"};
```

- (~) Inlined (clang only)

## STD::VARIANT - SUMMARY

- (+) Small/efficient memory footprint

- ```
Disconnect { };
Connecting { connection_id id{}; };
Connected  { connection_id id{};
            time last_ping{}; };
```

- (+) Integrates well with std::expected/static exceptions

- ```
return Error{"timeout"};
```

- (~) Inlined (clang only)
- (-) Hard to reuse (Similar to switch/enum)

# COROUTINES / LOOP - IMPLEMENTATION

# COROUTINES / LOOP - IMPLEMENTATION

```
auto Connection = [](auto& in) {  
    for (;;) { // Wait for an event... -> Disconnected  
        if (auto [event, data] = co_await in; event == connect) {  
            establish();
```

```
        } // Otherwise go back to co_await and suspend...  
    }  
};
```



# COROUTINES / LOOP - IMPLEMENTATION

```
auto Connection = [] (auto& in) {  
    for (;;) { // Wait for an event... -> Disconnected  
        if (auto [event, data] = co_await in; event == connect) {  
            establish();
```

```
        for (;;) { // -> Connecting  
            if (auto [event, data] = co_await in; event == established) {
```

```
        } end;;
```

```
    } // Otherwise go back to co_await and suspend...  
}  
};
```

# COROUTINES / LOOP - IMPLEMENTATION

```
auto Connection = [] (auto& in) {  
    for (;;) { // Wait for an event... -> Disconnected  
        if (auto [event, data] = co_await in; event == connect) {  
            establish();
```

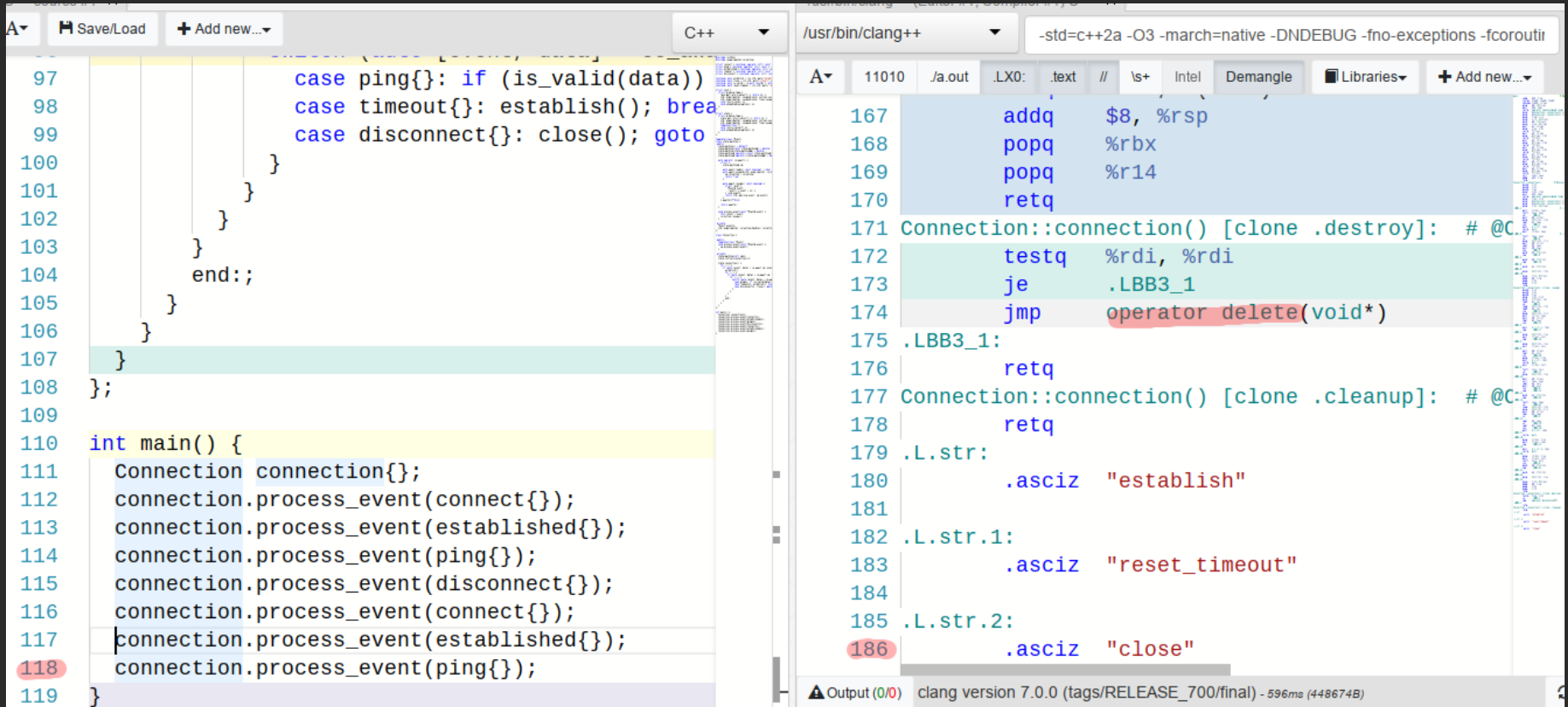
```
        for (;;) { // -> Connecting  
            if (auto [event, data] = co_await in; event == established) {
```

```
                for (;;) { // -> Connected  
                    switch (auto [event, data] = co_await in; event) {  
                        case ping:  
                            if (is_valid(data)) { reset_timeout(); continue; }  
                        case timeout: establish(); break;  
                        case disconnect: close(); goto end;  
                    }  
                }  
            }  
        }
```

```
    } end;;
```

```
    } // Otherwise go back to co_await and suspend...  
}  
};
```

# COROUTINES / LOOP - FULL EXAMPLE



The image shows a C++ IDE with two panels. The left panel displays the source code of a program, and the right panel displays the assembly output generated by the compiler.

**Source Code (Left Panel):**

```
97     case ping{}: if (is_valid(data))
98     case timeout{}: establish(); break;
99     case disconnect{}: close(); goto LBB3_1;
100 }
101 }
102 }
103 }
104 end;;
105 }
106 }
107 }
108 };
109
110 int main() {
111     Connection connection{};
112     connection.process_event(connect{});
113     connection.process_event(established{});
114     connection.process_event(ping{});
115     connection.process_event(disconnect{});
116     connection.process_event(connect{});
117     connection.process_event(established{});
118     connection.process_event(ping{});
119 }
```

**Assembly Output (Right Panel):**

```
11010 /a.out .LX0: .text // ls+ Intel Demangle Libraries + Add new...
167     addq    $8, %rsp
168     popq    %rbx
169     popq    %r14
170     retq
171 Connection::connection() [clone .destroy]: # @C
172     testq   %rdi, %rdi
173     je      .LBB3_1
174     jmp     operator delete(void*)
175 .LBB3_1:
176     retq
177 Connection::connection() [clone .cleanup]: # @C
178     retq
179 .L.str:
180     .asciz  "establish"
181
182 .L.str.1:
183     .asciz  "reset_timeout"
184
185 .L.str.2:
186     .asciz  "close"
```

**Output (Bottom Panel):**

```
Output (0/0) clang version 7.0.0 (tags/RELEASE_700/final) - 596ms (448674B)
```

[HTTPS://GODBOLT.ORG/Z/P3ZANT](https://godbolt.org/z/p3zant)

# COROUTINES / LOOP - SUMMARY

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)
- (~) Requires heap (heap elision / devirtualization)



## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)
- (~) Requires heap (heap elision / devirtualization)
- (~) Implicit states (position in the function)

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)
- (~) Requires heap (heap elision / devirtualization)
- (~) Implicit states (position in the function)
- (-) Events require a common type

## COROUTINES / LOOP - SUMMARY

- (+) Structured code using C++ features
- (+) Easily to switch between Async/Sync versions
- (~) Learning curve (different way of thinking)
- (~) Requires heap (heap elision / devirtualization)
- (~) Implicit states (position in the function)
- (-) Events require a common type
- (-) Weird usage of infinite loops

# COROUTINES / GOTO - IMPLEMENTATION

# COROUTINES / GOTO - IMPLEMENTATION

```
auto Connection = [] (auto& in) {  
    for (;;) {  
        disconnected: // State is represented by a position in the function  
        if (auto [event, data] = co_await in; event == connect) {  
            establish();  
        }  
    }  
};
```

```
} } };
```

# COROUTINES / GOTO - IMPLEMENTATION

```
auto Connection = [] (auto& in) {  
    for (;;) {  
        disconnected: // State is represented by a position in the function  
        if (auto [event, data] = co_await in; event == connect) {  
            establish();
```

```
            connecting:  
            if (auto [event, data] = co_await in; event == established) {
```

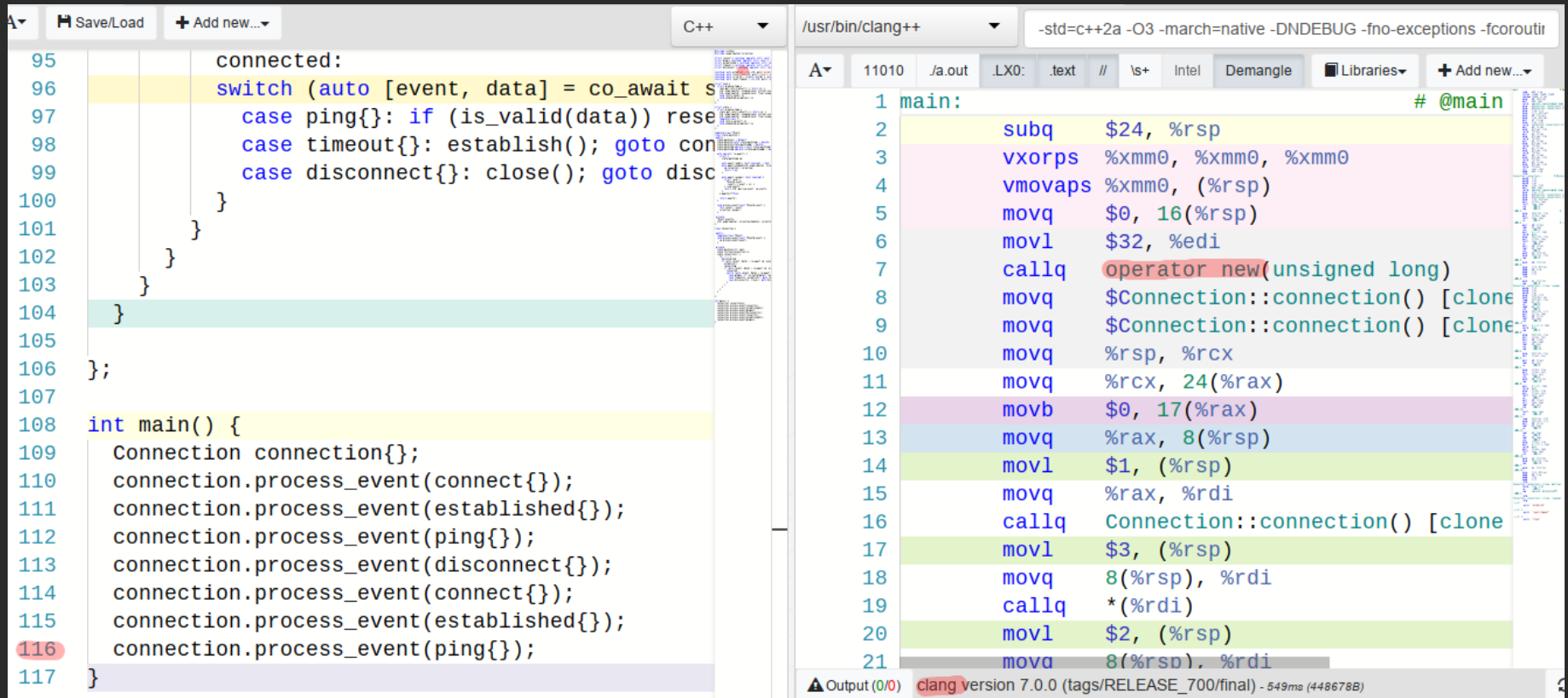
```
        }
```

```
    } } };
```

# COROUTINES / GOTO - IMPLEMENTATION

```
auto Connection = [] (auto& in) {  
    for (;;) {  
        disconnected: // State is represented by a position in the function  
        if (auto [event, data] = co_await in; event == connect) {  
            establish();  
  
            connecting:  
            if (auto [event, data] = co_await in; event == established) {  
  
                connected:  
                switch (auto [event, data] = co_await in; event) {  
                    case ping:  
                        if (is_valid(data)) { reset_timeout(); goto connected; }  
                    case timeout: establish();  
                        goto connecting; // Set the new state  
                    case disconnect:  
                        close(); goto disconnected;  
                }  
  
            }  
  
        }  
  
    } } };
```

# COROUTINES / GOTO - FULL EXAMPLE



The screenshot displays a C++ IDE with two panels. The left panel shows the source code, and the right panel shows the generated assembly code.

**Source Code (Left Panel):**

```
95     connected:
96     switch (auto [event, data] = co_await S
97         case ping{}: if (is_valid(data)) rese
98         case timeout{}: establish(); goto con
99         case disconnect{}: close(); goto disc
100    }
101    }
102    }
103    }
104    }
105    };
106    };
107    };
108    int main() {
109        Connection connection{};
110        connection.process_event(connect{});
111        connection.process_event(established{});
112        connection.process_event(ping{});
113        connection.process_event(disconnect{});
114        connection.process_event(connect{});
115        connection.process_event(established{});
116        connection.process_event(ping{});
117    }
```

**Assembly Code (Right Panel):**

```
1 main:                                     # @main
2     subq    $24, %rsp
3     vxorps  %xmm0, %xmm0, %xmm0
4     vmovaps %xmm0, (%rsp)
5     movq    $0, 16(%rsp)
6     movl    $32, %edi
7     callq   operator new(unsigned long)
8     movq    $Connection::connection() [clone
9     movq    $Connection::connection() [clone
10    movq    %rsp, %rcx
11    movq    %rcx, 24(%rax)
12    movb    $0, 17(%rax)
13    movq    %rax, 8(%rsp)
14    movl    $1, (%rsp)
15    movq    %rax, %rdi
16    callq   Connection::connection() [clone
17    movl    $3, (%rsp)
18    movq    8(%rsp), %rdi
19    callq   *(%rdi)
20    movl    $2, (%rsp)
21    movq    8(%rsp), %rdi
```

**Output (Bottom Panel):**

```
clang version 7.0.0 (tags/RELEASE_700/final) - 549ms (448678B)
```

[HTTPS://GODBOLT.ORG/Z/BJUHL9](https://godbolt.org/z/BJUHL9)



# COROUTINES / GOTO - SUMMARY

## COROUTINES / GOTO - SUMMARY

- (+) No infinite loops

## COROUTINES / GOTO - SUMMARY

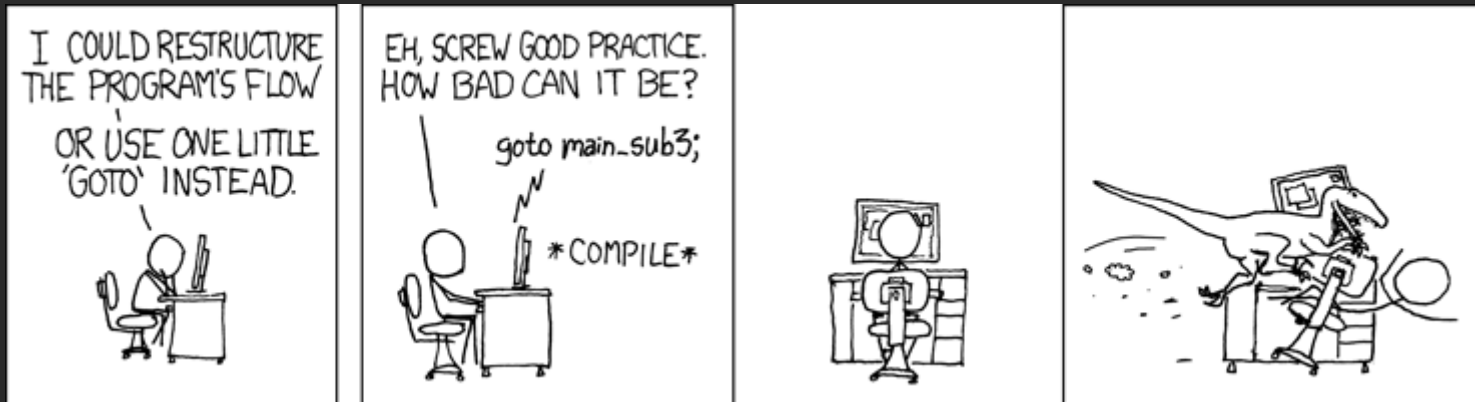
- (+) No infinite loops
- (~) Explicit states

## COROUTINES / GOTO - SUMMARY

- (+) No infinite loops
- (~) Explicit states
- (-) GOTO!

## COROUTINES / GOTO - SUMMARY

- (+) No infinite loops
- (~) Explicit states
- (-) GOTO!



# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

$$\left. \begin{array}{l} \text{ } \end{array} \right\} \left. \begin{array}{l} \text{ } \end{array} \right\}$$

# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

```
auto disconnected() {  
    for (;;) { // Wait for the connect event...  
  
        if (auto const event = co_await in; std::get_if<connect>(&event)) {  
            establish(); co_return connecting(); // Set the new state  
        }  
  
    } }  
}
```



# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

```
auto disconnected() {  
    for (;;) { // Wait for the connect event...  
  
        if (auto const event = co_await in; std::get_if<connect>(&event)) {  
            establish(); co_return connecting(); // Set the new state  
        }  
  
    } }  
}
```

```
auto connected() {  
    for (;;) { // Wait for the ping event...  
        auto const event = co_await in;
```

```
    } }  
}
```

# COROUTINES / FUNCTIONS / VARIANT - IMPLEMENTATION

```
auto disconnected() {  
    for (;;) { // Wait for the connect event...
```

```
        if (auto const event = co_await in; std::get_if<connect>(&event)) {  
            establish(); co_return connecting(); // Set the new state  
        }
```

```
    } }
```

```
auto connected() {  
    for (;;) { // Wait for the ping event...  
        auto const event = co_await in;
```

```
        if (std::get_if<ping>(&event) and is_valid(std::get<ping>(event))) {  
            reset_timeout();  
        } else if (std::get_if<timeout>(&event)) {  
            establish(); co_return connecting();  
        } else if (std::get_if<disconnect>(&event)) {  
            close(); co_return disconnected();  
        }
```

```
    } }
```

# COROUTINES / FUNCTIONS / VARIANT - FULL EXAMPLE

The image shows a C++ IDE with two panels. The left panel displays C++ source code for a coroutine-based connection manager. The right panel shows the corresponding assembly code generated by Clang 7.0.0.

**Source Code (Left Panel):**

```
100 if (std::get_if<ping>(&event) and is_valid(
101     reset_timeout();
102 } else if (std::get_if<timeout>(&event)) {
103     establish();
104 co_return connecting();
105 } else if (std::get_if<disconnect>(&event))
106     close();
107 co_return disconnected();
108 }
109 }
110 }
111 };
112
113 int main() {
114     Connection connection{};
115     connection.process_event(connect{});
116     connection.process_event(established{});
117     connection.process_event(ping{});
118     connection.process_event(disconnect{});
119     connection.process_event(connect{});
120     connection.process_event(established{});
121     connection.process_event(ping{});
122 }
```

**Assembly Code (Right Panel):**

```
377 addq $8, %rsp
378 popq %rbx
379 popq %r14
380 retq
381 Connection::disconnected() [clone .destroy]: # @
382 testq %rdi, %rdi
383 je .LBB12_1
384 jmp operator delete(void*)
385 .LBB12_1:
386 retq
387 Connection::disconnected() [clone .cleanup]: # @
388 retq
389 .L.str:
390 .asciz "establish"
391
392 .L.str.1:
393 .asciz "reset_timeout"
394
395 .L.str.2:
396 .asciz "close"
```

**Output (Bottom):** clang version 7.0.0 (tags/RELEASE\_700/final) - 1273ms (1868986B)

[HTTPS://GODBOLT.ORG/Z/TCIWKH](https://godbolt.org/z/TCIWKH)

# COROUTINES / FUNCTIONS / VARIANT - SUMMARY

## COROUTINES / FUNCTIONS / VARIANT - SUMMARY

- (+) Easier to add/follow new states/behaviour

## COROUTINES / FUNCTIONS / VARIANT - SUMMARY

- (+) Easier to add/follow new states/behaviour
- (+) Type safe events

## COROUTINES / FUNCTIONS / VARIANT - SUMMARY

- (+) Easier to add/follow new states/behaviour
- (+) Type safe events



# BOOST



# BOOST

| Library  | Boost.Statechart | Boost.MSM   | [Boost].SML |
|----------|------------------|-------------|-------------|
| Standard | C++98/03         | C++98/03    | C++14       |
| Version  | 1.68             | 1.68        | 1.1.0       |
| License  | Boost 1.0        | Boost 1.0   | Boost 1.0   |
| Linkage  | header only      | header only | header only |
| UML      | 1.5              | 2.0         | 2.5         |

Disclaimer: [Boost].SML is not an official Boost library

# BOOST.STATECHART - IMPLEMENTATION

# BOOST.STATECHART - IMPLEMENTATION

## EVENTS

```
struct connect      : sc::event<connect> {};  
struct ping        : sc::event<ping> {};  
struct established : sc::event<established> {};  
struct timeout     : sc::event<timeout> {};  
struct disconnect  : sc::event<disconnect> {};
```

# BOOST.STATECHART - IMPLEMENTATION

## EVENTS

```
struct connect      : sc::event<connect> {};  
struct ping        : sc::event<ping> {};  
struct established : sc::event<established> {};  
struct timeout     : sc::event<timeout> {};  
struct disconnect  : sc::event<disconnect> {};
```

## ACTIONS/GUARDS

```
struct Connection : sc::state_machine<Connection, Disconnected> {  
    template<class TEvent>  
    void establish(TEvent const&) { std::puts("establish"); }  
    void reset_timeout(ping const&) { std::puts("reset_timeout"); }  
    void close(disconnect const&) { std::puts("close"); }  
    bool is_valid(ping const&) { return true; }  
};
```

# BOOST.STATECHART - IMPLEMENTATION

# BOOST.STATECHART - IMPLEMENTATION

```
struct Disconnected : sc::simple_state<Disconnected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<connect, Connecting,  
                        Connection, &Connection::establish>>;  
};
```

# BOOST.STATECHART - IMPLEMENTATION

```
struct Disconnected : sc::simple_state<Disconnected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<connect, Connecting,  
                        Connection, &Connection::establish>>;  
};
```

```
struct Connected : sc::simple_state<Connected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<timeout, Connecting,  
                        Connection, &Connection::establish>,  
        sc::transition<disconnect, Disconnected,  
                        Connection, &Connection::close>,  
        sc::custom_reaction<ping>  
    >;
```

# BOOST.STATECHART - IMPLEMENTATION

```
struct Disconnected : sc::simple_state<Disconnected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<connect, Connecting,  
            Connection, &Connection::establish>>;  
};
```

```
struct Connected : sc::simple_state<Connected, Connection> {  
    using reactions = mpl::list<  
        sc::transition<timeout, Connecting,  
            Connection, &Connection::establish>,  
        sc::transition<disconnect, Disconnected,  
            Connection, &Connection::close>,  
        sc::custom_reaction<ping>  
    >;
```

```
sc::result react(ping const& event) {  
    if (context<Connection>().is_valid(event)) {  
        context<Connection>().reset_timeout(event);  
    }  
    return discard_event();  
}
```



# BOOST.STATECHART - FULL EXAMPLE

The image displays a code editor with two panes. The left pane shows C++ source code for a Boost.Statechart example, and the right pane shows the corresponding assembly code generated by the compiler.

**C++ Source Code (Left Pane):**

```
40     sc::transition<disconnect, Disconnected, Conn
41     sc::custom_reaction<ping>
42     >;
43
44     sc::result react(ping const & event) {
45         if (context<Connection>().is_valid(event)) {
46             context<Connection>().reset_timeout(event);
47         }
48         return discard_event();
49     }
50 };
51
52 int main() {
53     Connection connection{};
54     connection.initiate();
55     connection.process_event(connect{});
56     connection.process_event(established{});
57     connection.process_event(ping{});
58     connection.process_event(disconnect{});
59     connection.process_event(connect{});
60     connection.process_event(established{});
61     connection.process_event(ping{});
62 }
```

**Assembly Output (Right Pane):**

The right pane shows the assembly code generated by the compiler. The top section is for the `react` function, and the bottom section is for the `main` function.

**react Function Assembly:**

```
43     movl    $1, %esi
44     movq    %r14, %rdi
45     vzeroupper
46     callq   boost::statechart::state_machine
47     movq    $0, 200(%rsp)
48     movl    $48, %edi
49     callq   operator new(unsigned long)
50     movq    %rax, %rbx
```

**main Function Assembly:**

```
2854     .quad   typeid for Connected
2855     .quad   boost::statechart::simple_state<
2856     .quad   Connected::~~Connected() [comple
2857     .quad   Connected::~~Connected() [deletin
2858     .quad   boost::statechart::simple_state<
2859     .quad   boost::statechart::detail::leaf
2860     .quad   boost::statechart::simple_state<
```

[HTTPS://GODBOLT.ORG/Z/NN8UYH](https://godbolt.org/z/NN8UYH)

# BOOST.STATECHART - SUMMARY

## BOOST.STATECHART - SUMMARY

- (+) UML-1.5 features

## BOOST.STATECHART - SUMMARY

- (+) UML-1.5 features
- (~) Learning curve (Similar to State Pattern)

## BOOST.STATECHART - SUMMARY

- (+) UML-1.5 features
- (~) Learning curve (Similar to State Pattern)
- (-) Dynamic allocations

## BOOST.STATECHART - SUMMARY

- (+) UML-1.5 features
- (~) Learning curve (Similar to State Pattern)
- (-) Dynamic allocations
- (-) Dynamic dispatch

## BOOST.STATECHART - SUMMARY

- (+) UML-1.5 features
- (~) Learning curve (Similar to State Pattern)
- (-) Dynamic allocations
- (-) Dynamic dispatch
- (-) High memory footprint

# BOOST.MSM / EUML - IMPLEMENTATION



# BOOST.MSM / EUML - IMPLEMENTATION

## EVENTS

```
BOOST_MSM_EUML_EVENT(connect)
BOOST_MSM_EUML_EVENT(ping)
BOOST_MSM_EUML_EVENT(established)
BOOST_MSM_EUML_EVENT(timeout)
BOOST_MSM_EUML_EVENT(disconnect)
```

# BOOST.MSM / EUML - IMPLEMENTATION

## EVENTS

```
BOOST_MSM_EUML_EVENT(connect)
BOOST_MSM_EUML_EVENT(ping)
BOOST_MSM_EUML_EVENT(established)
BOOST_MSM_EUML_EVENT(timeout)
BOOST_MSM_EUML_EVENT(disconnect)
```

## STATES

```
BOOST_MSM_EUML_STATE((), Disconnected)
BOOST_MSM_EUML_STATE((), Connecting)
BOOST_MSM_EUML_STATE((), Connected)
```

# BOOST.MSM / EUML - IMPLEMENTATION

# BOOST.MSM / EUML - IMPLEMENTATION

## ACTIONS

```
BOOST_MSM_EUML_ACTION(establish) {  
    template <class FSM, class EVT, class SourceState, class TargetState>  
    void operator()(EVT const&, FSM &, SourceState &, TargetState &) {  
        std::puts("establish");  
    }  
};
```

# BOOST.MSM / EUML - IMPLEMENTATION

## ACTIONS

```
BOOST_MSM_EUML_ACTION(establish) {  
    template <class FSM, class EVT, class SourceState, class TargetState>  
    void operator()(EVT const&, FSM &, SourceState &, TargetState &) {  
        std::puts("establish");  
    }  
};
```

## GUARDS

```
BOOST_MSM_EUML_ACTION(is_valid) {  
    template <class FSM, class EVT, class SourceState, class TargetState>  
    auto operator()(EVT const&, FSM&, SourceState&, TargetState&) {  
        return true;  
    }  
};
```

# BOOST.MSM / EUML - IMPLEMENTATION

## BOOST.MSM / EUML - IMPLEMENTATION

```
BOOST_MSM_EUML_TRANSITION_TABLE((  
    Connecting    == Disconnected + connect / establish,  
    Connected     == Connecting   + established,  
                  Connected      + ping [ is_valid ] / reset_timeout,  
    Connecting    == Connected   + timeout / establish,  
    Disconnected == Connected   + disconnect / close  
),  
transition_table)
```

## BOOST.MSM / EUML - IMPLEMENTATION

```
BOOST_MSM_EUML_TRANSITION_TABLE((
    Connecting    == Disconnected + connect / establish,
    Connected     == Connecting    + established,
                  Connected      + ping [ is_valid ] / reset_timeout,
    Connecting    == Connected    + timeout / establish,
    Disconnected == Connected     + disconnect / close
),
transition_table)
```

```
BOOST_MSM_EUML_DECLARE_STATE_MACHINE
    ((transition_table, init_ << Disconnected), ConnectionImpl)
```



## BOOST.MSM / EUML - IMPLEMENTATION

```
BOOST_MSM_EUML_TRANSITION_TABLE((  
    Connecting    == Disconnected + connect / establish,  
    Connected     == Connecting   + established,  
                  Connected      + ping [ is_valid ] / reset_timeout,  
    Connecting    == Connected    + timeout / establish,  
    Disconnected == Connected    + disconnect / close  
),  
transition_table)
```

```
BOOST_MSM_EUML_DECLARE_STATE_MACHINE  
    ((transition_table, init_ << Disconnected), ConnectionImpl)
```

```
using Connection = msm::back::state_machine<ConnectionImpl>;
```

# BOOST.MSM / EURL - FULL EXAMPLE

The screenshot displays a C++ IDE with two main panels. The left panel shows the source code for a Boost.MSM state machine. The right panel shows the assembly output generated by Clang and GCC.

**Source Code (Left Panel):**

```
59 BOOST_MSM_EURL_DECLARE_STATE_MACHINE((
60     transition_table,
61     init_ << Disconnected,
62     no_action,
63     no_action,
64     attributes_ << no_attributes_,
65     configure_ << no_exception << no_msg_queue,
66     Log_No_Transition
67 ), ConnectionImpl)
68
69 using Connection = msm::back::state_machine<Conne
70
71 int main() {
72     Connection connection{};
73     connection.start();
74     connection.process_event(connect);
75     connection.process_event(established);
76     connection.process_event(ping);
77     connection.process_event(disconnect);
78     connection.process_event(connect);
79     connection.process_event(established);
80     connection.process_event(ping);
81 }
```

**Assembly Output (Right Panel):**

**clang version 7.0.0 (tags/RELEASE\_700/final) - 169ms (6903748B)**

```
11010 /a.out LX0: .text // ls+ Intel Demangle Libraries+ Add new...
21 movl $$_ZL7connect, %ecx
22 movq %rbx, %rdi
23 callq *boost::msm::back::dispatch_table
24 movslq 4(%rsp), %rdx
25 xorl %esi, %esi
26 movl $$_ZL11established, %ecx
27 movq %rbx, %rdi
28 callq *boost::msm::back::dispatch_table
```

**g++ (GCC) 8.2.0 - 140ms (4449476B)**

```
191 .zero 32
192 boost::msm::back::dispatch_table<boost::msm::bac
193 .zero 32
194 .LC3:
195 .quad boost::msm::back::HandledEnum bo
196 .LC4:
197 .quad boost::msm::back::HandledEnum bo
```

[HTTPS://GODBOLT.ORG/Z/NVTVOJ](https://godbolt.org/z/NVTVOJ)

# BOOST.MSM / EUML - SUMMARY

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch  $O(1)$  - jump table

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch  $O(1)$  - jump table
- (+) UML-2.0 features

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch  $O(1)$  - jump table
- (+) UML-2.0 features
- (+) Small memory footprint

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch  $O(1)$  - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve



## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch  $O(1)$  - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve
- (~) DSL based

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch  $O(1)$  - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve
- (~) DSL based
- (-) Macro based

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch  $O(1)$  - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve
- (~) DSL based
- (-) Macro based
- (-) Slow compilation times
  - Timeouts in the Compiler-Explorer

## BOOST.MSM / EUML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Dispatch  $O(1)$  - jump table
- (+) UML-2.0 features
- (+) Small memory footprint
- (~) Learning curve
- (~) DSL based
- (-) Macro based
- (-) Slow compilation times
  - Timeouts in the Compiler-Explorer
- (-) Error messages

# **[BOOST].SML - IMPLEMENTATION**

## [BOOST].SML - IMPLEMENTATION

```
sml::sm connection = []{  
    using namespace sml;  
    return transition_table{  
        * "Disconnected"_s + event<connect> / establish      = "Connecting"_s,  
          "Connecting"_s   + event<established>              = "Connected"_s,  
          "Connected"_s    + event<ping> [ is_valid ] / reset_timeout,  
          "Connected"_s    + event<timeout> / establish      = "Connecting"_s,  
          "Connected"_s    + event<disconnect> / close       = "Disconnected"_s  
    };  
};
```

# [BOOST].SML - PERFORMANCE TUNING



# [BOOST].SML - PERFORMANCE TUNING

*Don't pay for what you don't use!*





## DISPATCH POLICY CHANGES EVENT DISPATCHING STRATEGY

| Name             | Policy                  | Default                           |
|------------------|-------------------------|-----------------------------------|
| Jump Table       | <code>jump_table</code> | <code>gcc &lt; 8.0</code>         |
| Nested Switch    | <code>switch</code>     | <code>gcc &gt;= 8.0</code>        |
| If/Else          | <code>branch</code>     | <code>clang</code>                |
| Fold expressions | <code>fold_expr</code>  | <code>gcc/clang with C++17</code> |

## [BOOST].SML / DISPATCH POLICY - IF/ELSE

## [BOOST].SML / DISPATCH POLICY - IF/ELSE

```
template <class TMappings, // back-end -> generated transitions
    auto N,
    class TState,
    class... TStates,
    class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

}
```

## [BOOST].SML / DISPATCH POLICY - IF/ELSE

```
template <class TMappings, // back-end -> generated transitions
        auto N,
        class TState,
        class... TStates,
        class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

    if constexpr(sizeof...(TStates...) > 0) {
        return current_state == N
            ? TMappings<TState>::execute(event)
            : dispatch<TMappings, N + 1, TStates...>(current_state, event);
    }

}
```

# [BOOST].SML / DISPATCH POLICY - IF/ELSE

The screenshot displays a C++ IDE with two panels. The left panel shows the source code for a state machine library, and the right panel shows the assembly output for two different compilers.

**Source Code (Left Panel):**

```
20 return make_transition_table(  
21     * "Disconnected"_s + event<connect> / estab  
22     "Connecting"_s    + event<established>  
23     "Connected"_s     + event<ping> [ is_valid  
24     "Connected"_s     + event<timeout> / estab  
25     "Connected"_s     + event<disconnect> / cl  
26 );  
27 }  
28 ;  
29  
30 nt main() {  
31     sml::sm<Connection,  
32     sml::dispatch<sml::back::policies::branch_stm  
33     connection.process_event(connect{});  
34     connection.process_event(established{});  
35     connection.process_event(ping{});  
36     connection.process_event(disconnect{});  
37     connection.process_event(connect{});  
38     connection.process_event(established{});  
39     connection.process_event(ping{});  
40 }
```

**Assembly Output (Right Panel):**

The right panel shows the assembly output for two compilers. The top panel is for Clang version 7.0.0, and the bottom panel is for g++ (GCC) 8.2.0.

**Clang Assembly Output:**

```
1 main:                                # @main  
2     pushq    %rax  
3     movl     $.L.str, %edi  
4     callq    puts  
5     movl     $.L.str.1, %edi  
6     callq    puts  
7     movl     $.L.str.2, %edi
```

**g++ Assembly Output:**

```
15     mov     edi, OFFSET FLAT:.LC0  
16     call    puts  
17     mov     edi, OFFSET FLAT:.LC1  
18     call    puts  
19     xor     eax, eax  
20     add     rsp, 8  
21     ret
```

[HTTPS://GODBOLT.ORG/Z/DSN1PF](https://godbolt.org/z/DSN1PF)

## **[BOOST].SML / DISPATCH POLICY - SWITCH**

## [BOOST].SML / DISPATCH POLICY - SWITCH

```
template <class TMappings, // back-end -> generated transitions
    auto N = 0,
    class TState,
    class... TStates,
    class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

}
```

## [BOOST].SML / DISPATCH POLICY - SWITCH

```
template <class TMappings, // back-end -> generated transitions
    auto N = 0,
    class TState,
    class... TStates,
    class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

    if constexpr(sizeof...(TStates...) > 0) {
        switch (current_state) {
            default: return dispatch<TMappings, N + 1, TStates...>(
                current_state, event);
            case N:  return TMappings<TState>::execute(event);
        }
    }
}
```



# [BOOST].SML / DISPATCH POLICY - SWITCH

The image shows a code editor with two panes. The left pane displays C++ source code for a Boost.SML dispatcher using a switch policy. The right pane shows the assembly output for the same code, compiled with clang++ and gcc++.

```
17 struct Connection {
18     auto operator>()() const {
19         using namespace sml;
20         return make_transition_table(
21             * "Disconnected"_s + event<connect> / estab
22             "Connecting"_s    + event<established>
23             "Connected"_s     + event<ping> [ is_valid
24             "Connected"_s     + event<timeout> / estab
25             "Connected"_s     + event<disconnect> / cl
26         );
27     }
28 };
29
30 int main() {
31     sml::sm<Connection,
32         sml::dispatch<sml::back::policies::switch_str
33     connection.process_event(connect{});
34     connection.process_event(established{});
35     connection.process_event(ping{});
36     connection.process_event(disconnect{});
37     connection.process_event(connect{});
38     connection.process_event(established{});
39     connection.process_event(ping{});
40 }
```

Assembly output (clang++):

```
1 main:                                     # @main
2     pushq   %rax
3     movl    $.L.str, %edi
4     callq   puts
5     movl    $.L.str.1, %edi
6     callq   puts
7     movl    $.L.str.2, %edi
8     callq   puts
```

Assembly output (gcc++):

```
14     call    puts
15     mov     edi, OFFSET FLAT:.LC0
16     call    puts
17     mov     edi, OFFSET FLAT:.LC1
18     call    puts
19     xor     eax, eax
20     add     rsp, 8
21     ret
```

[HTTPS://GODBOLT.ORG/Z/AKMGiy](https://godbolt.org/z/AKMGiy)

## [BOOST].SML / DISPATCH POLICY - JUMP TABLE

## [BOOST].SML / DISPATCH POLICY - JUMP TABLE

```
template <class TMappings, // back-end -> generated transitions
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

}
```

## [BOOST].SML / DISPATCH POLICY - JUMP TABLE

```
template <class TMappings, // back-end -> generated transitions
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

    using dispatch_table_t = bool (*) (TEvent const&);

    constexpr static dispatch_table_t dispatch_table[] = {
        &TMappings<TStates>::template execute<TEvent>...
    };

    return dispatch_table[current_state] (event);

}
```

# [BOOST].SML / DISPATCH POLICY - JUMP TABLE

The image shows a C++ IDE with two panels. The left panel displays the source code for a Boost.SML dispatch policy jump table. The right panel shows the assembly output for the same code, compiled with clang++ and g++.

```
17 struct Connection {
18     auto operator>()() const {
19         using namespace sml;
20         return make_transition_table(
21             * "Disconnected"_s + event<connect> / establ
22             "Connecting"_s + event<established>
23             "Connected"_s + event<ping> [ is_valid
24             "Connected"_s + event<timeout> / establ
25             "Connected"_s + event<disconnect> / clc
26         );
27     }
28 ;
29
30 int main() {
31     sml::sm<Connection,
32         sml::dispatch<sml::back::policies::jump_table>
33     connection.process_event(connect{});
34     connection.process_event(established{});
35     connection.process_event(ping{});
36     connection.process_event(disconnect{});
37     connection.process_event(connect{});
38     connection.process_event(established{});
39     connection.process_event(ping{});
40 }
```

The right panel shows the assembly output for the clang++ compiler. The assembly code is as follows:

```
39 bool boost::sml::v1_1_0::back::transitions<boost
40     pushq    %rax
41     movb     $1, (%r8)
42     movl     $.L.str, %edi
43     callq    puts
44     movb     $1, %al
45     popq     %rcx
```

The right panel also shows the assembly output for the g++ compiler. The assembly code is as follows:

```
112     .quad    bool boost::sml::v1_1_0::back::t
113     .quad    bool boost::sml::v1_1_0::back::t
114 bool boost::sml::v1_1_0::back::policies::jump_ta
115     .quad    _ZN5boost3sml6v1_1_04back11trans
116     .quad    bool boost::sml::v1_1_0::back::t
117     .quad    bool boost::sml::v1_1_0::back::t
118     .quad    bool boost::sml::v1_1_0::back::t
```

[HTTPS://GODBOLT.ORG/Z/LVJIX9](https://godbolt.org/z/LVJIX9)

## [BOOST].SML / DISPATCH POLICY - FOLD EXPRESSIONS (C++17)

## [BOOST].SML / DISPATCH POLICY - FOLD EXPRESSIONS (C++17)

```
template <class TMappings, // back-end -> generated transitions
          auto... Ns,
          class... TStates,
          class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

}
```

## [BOOST].SML / DISPATCH POLICY - FOLD EXPRESSIONS (C++17)

```
template <class TMappings, // back-end -> generated transitions
         auto... Ns,
         class... TStates,
         class TEvent>
constexpr auto dispatch(state_t &current_state, TEvent const &event) {

    static_assert(sizeof...(TStates) == sizeof...(Ns));

    return ((
        current_state == Ns
        ? TMappings<TStates>::execute(event)
        : false
    ) or ...);

}
```



# [BOOST].SML / DISPATCH POLICY - FOLD EXPRESSIONS (C++17)

The image shows a C++ IDE with two panes. The left pane displays the source code for a program using Boost.SML. The right pane shows the assembly output generated by the compiler.

**Source Code (Left Pane):**

```
17 struct Connection {
18     auto operator>()() const {
19         using namespace sml;
20         return make_transition_table(
21             * "Disconnected"_s + event<connect> / establ
22             "Connecting"_s + event<established>
23             "Connected"_s + event<ping> [ is_valid
24             "Connected"_s + event<timeout> / establ
25             "Connected"_s + event<disconnect> / clo
26         );
27     }
28 ;
29
30 int main() {
31     sml::sm<Connection,
32         sml::dispatch<sml::back::policies::fold_expr>>
33     connection.process_event(connect{});
34     connection.process_event(established{});
35     connection.process_event(ping{});
36     connection.process_event(disconnect{});
37     connection.process_event(connect{});
38     connection.process_event(established{});
39     connection.process_event(ping{});
```

**Assembly Output (Right Pane):**

The assembly output is divided into two sections. The top section is for the `main` function, generated by `clang version 7.0.0`. The bottom section is for the `main` function, generated by `g++ (GCC) 8.2.0`.

**clang assembly (Top):**

```
1 main:                                     # @main
2     pushq   %rax
3     movl    $.L.str, %edi
4     callq   puts
5     movl    $.L.str.1, %edi
6     callq   puts
7     movl    $.L.str.2, %edi
8     callq   puts
```

**g++ assembly (Bottom):**

```
14     call    puts
15     mov     edi, OFFSET FLAT:.LC0
16     call    puts
17     mov     edi, OFFSET FLAT:.LC1
18     call    puts
19     xor     eax, eax
20     add     rsp, 8
21     ret
```

[HTTPS://GODBOLT.ORG/Z/V\\_B7NM](https://godbolt.org/z/V_B7NM)

# [BOOST].SML - SUMMARY

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch  $O(1)$

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch  $O(1)$
- (+) Fast compilation times

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch  $O(1)$
- (+) Fast compilation times
- (+) UML-2.5 features

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch  $O(1)$
- (+) Fast compilation times
- (+) UML-2.5 features
- (+) Minimal memory footprint
  - `sizeof(Connection) == 1b`



## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch  $O(1)$
- (+) Fast compilation times
- (+) UML-2.5 features
- (+) Minimal memory footprint
  - `sizeof(Connection) == 1b`
- (~) Learning curve

## [BOOST].SML - SUMMARY

- (+) Declarative/Expressive (UML transition)
- (+) Customizable (At compile time)
- (+) Inlined / Dispatch  $O(1)$
- (+) Fast compilation times
- (+) UML-2.5 features
- (+) Minimal memory footprint
  - `sizeof(Connection) == 1b`
- (~) Learning curve
- (~) DSL based

# SOLUTIONS - SUMMARY

# SOLUTIONS - SUMMARY

| Solution            | State representation | Transition Table | Transition    |
|---------------------|----------------------|------------------|---------------|
| Naive - If/Else     | Boolean              | Per State        | Implicit      |
| Naive - Switch/Enum | Enum                 | Per State        | Implicit      |
| Naive - Inheritance | Class                | Per State        | Implicit      |
| STL - std::variant  | Union                | Per State        | Implicit      |
| STL - coroutines    | Function             | Global           | Implicit      |
| Boost.Statechart    | Class                | Per State        | Semi-explicit |
| Boost.MSM           | Class                | Global           | Explicit      |
| Boost.SML           | Type                 | Global           | Explicit      |

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| Per State/Global  | Transitions described per state/for all states at once     |
| Implicit/Explicit | Transition hidden/visible directly on the transition table |

# BENCHMARKS

# ENVIRONMENT / SETUP

# ENVIRONMENT / SETUP

```
const auto action/guard = []{  
    asm volatile("" : : : "memory");  
};
```

# ENVIRONMENT / SETUP

```
const auto action/guard = []{  
    asm volatile("" : : : "memory");  
};
```

```
int main() {  
    constexpr auto size = 1'000'000;  
    std::array events = rand_events<size>();  
  
    Connection connection{};  
    for (auto i = 0; i < size; ++i) {  
        process_event(events[i]);  
    }  
}
```



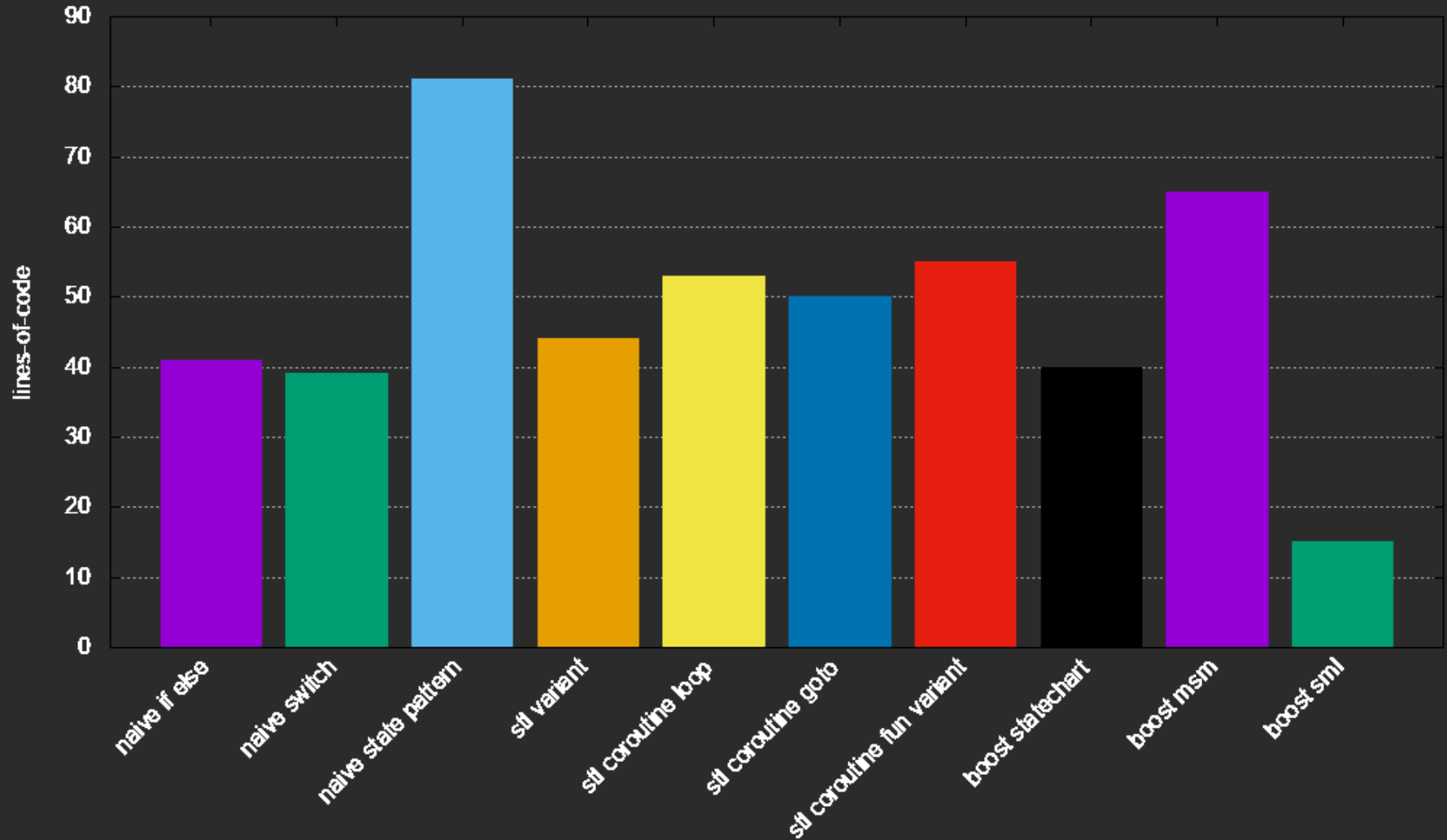
# ENVIRONMENT / SETUP

```
const auto action/guard = []{  
    asm volatile("" : : : "memory");  
};
```

```
int main() {  
    constexpr auto size = 1'000'000;  
    std::array events = rand_events<size>();  
  
    Connection connection{};  
    for (auto i = 0; i < size; ++i) {  
        process_event(events[i]);  
    }  
}
```

```
$CXX -std=c++2a      # clang-7.0.0/gcc-8.2  
-stdlib=libc++      # clang-7.0.0  
-fcoroutines-ts     # clang-7.0.0  
-O3 -march=native -flto -fno-exceptions -DNDEBUG  
-I boost_1_68_0
```

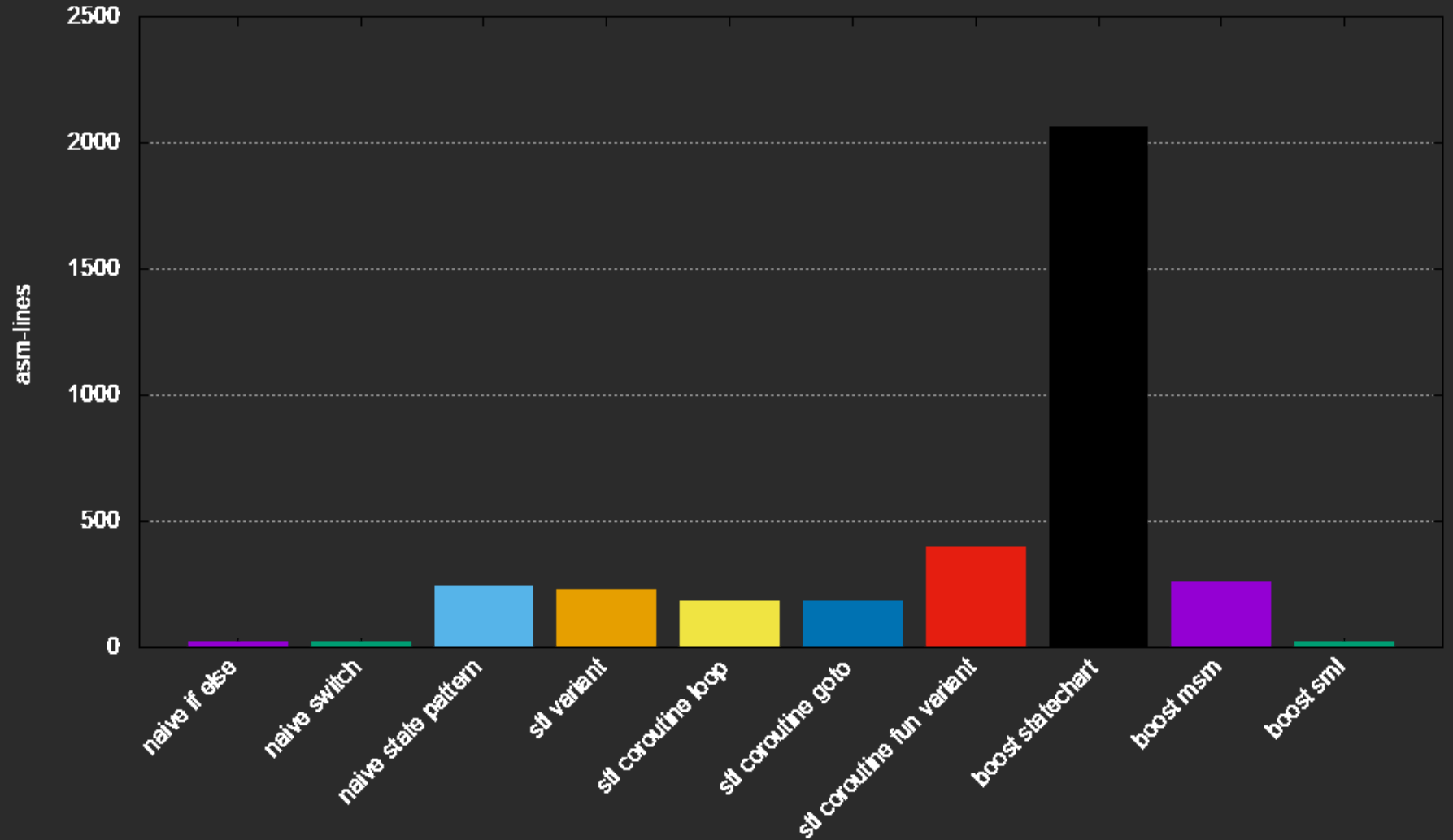
# BENCHMARKS - LINES OF CODE (LOC)



(less is better)



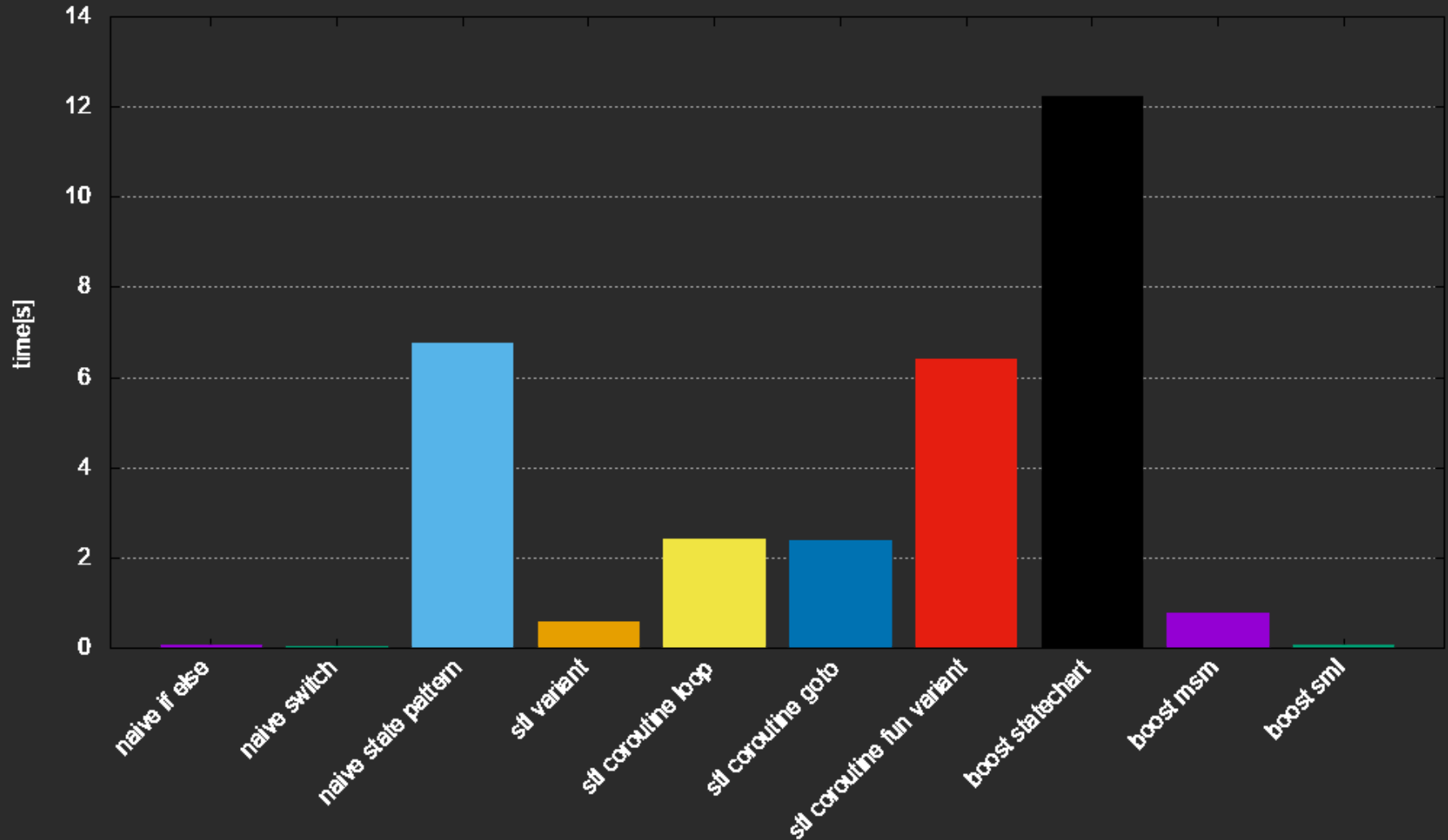
# BENCHMARKS - ASM LINES



(less is better)



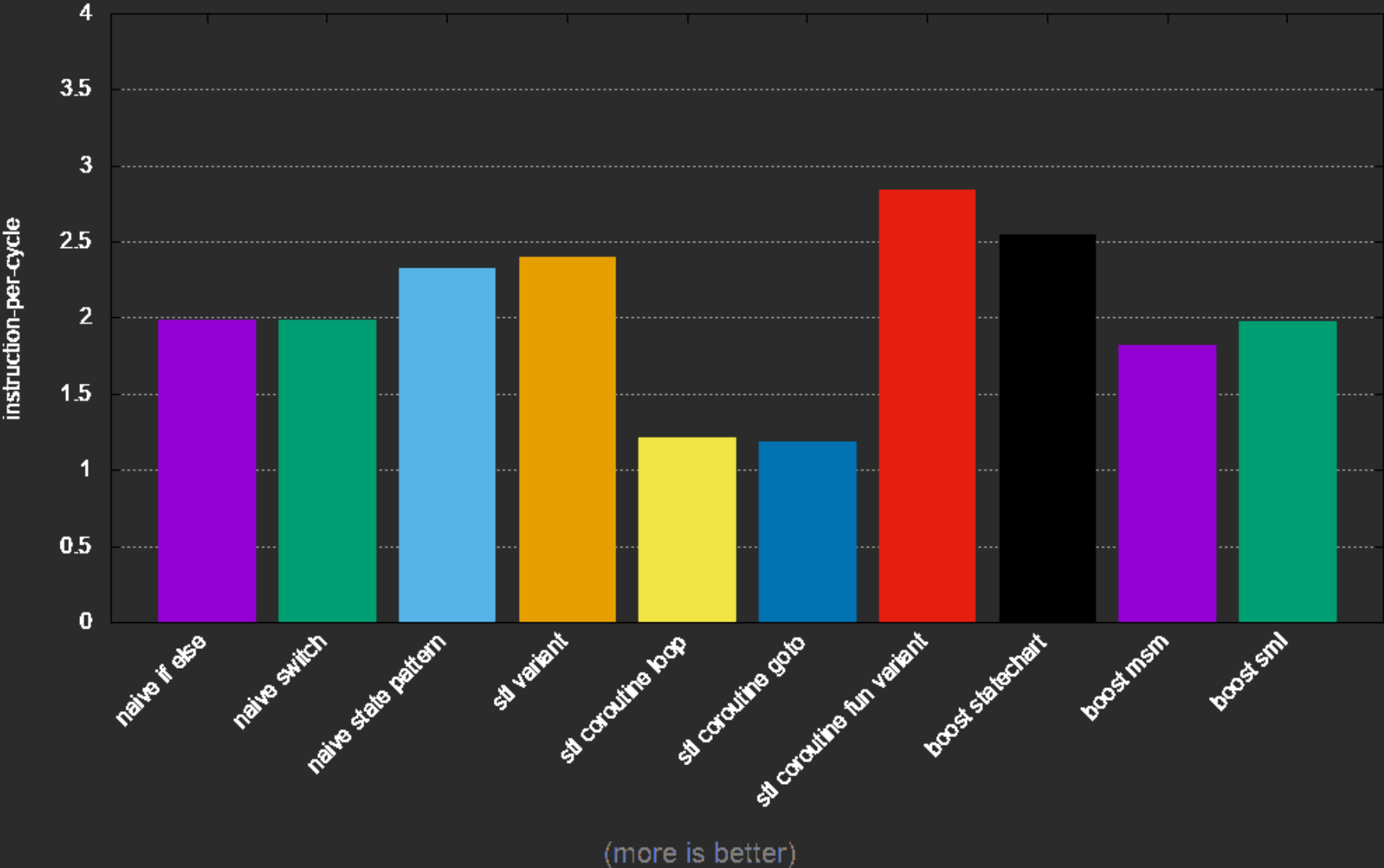
# BENCHMARKS - RUN-TIME PERFORMANCE - TIME / RELEASE



(less is better)



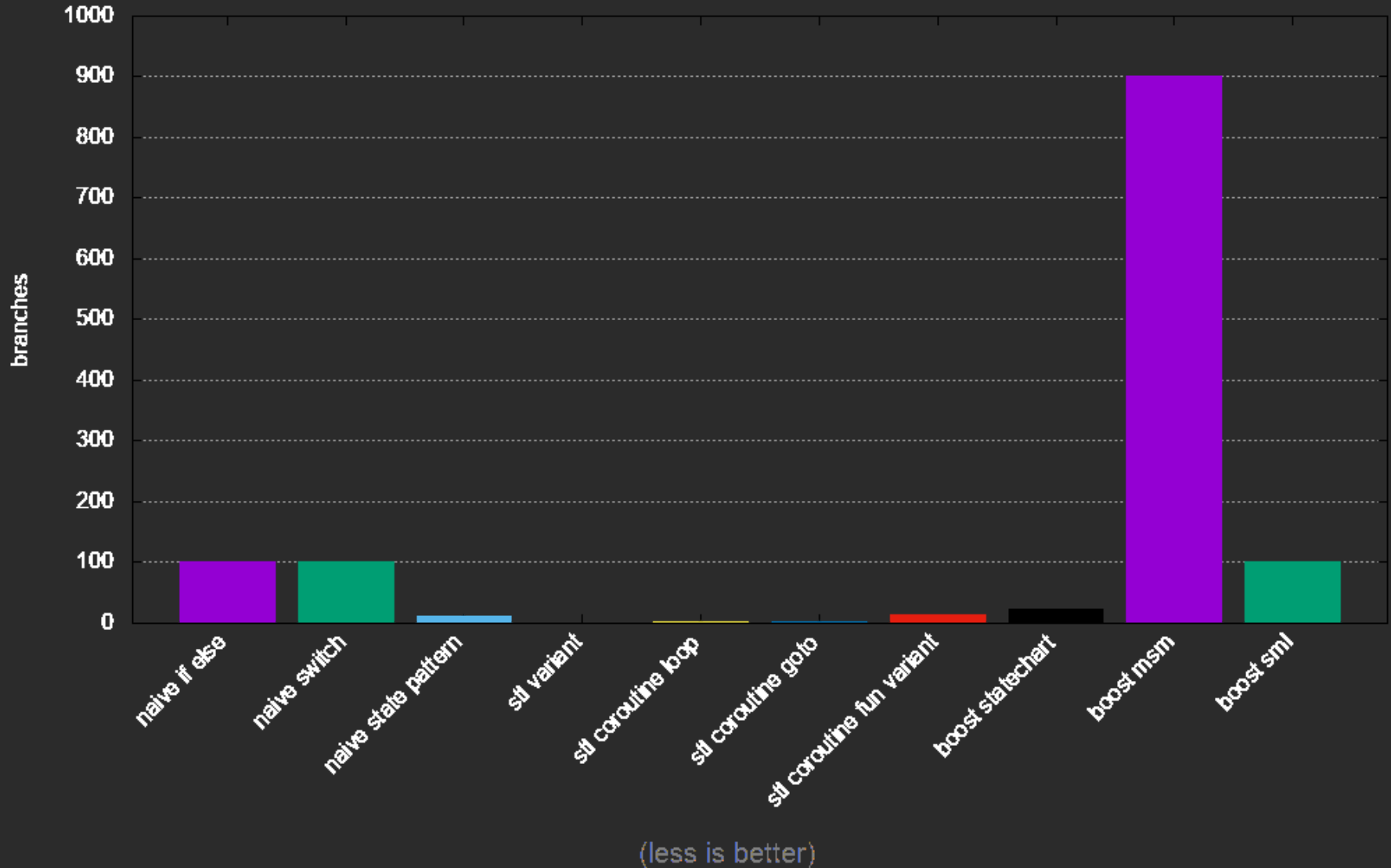
# BENCHMARKS - RUN-TIME PERFORMANCE - INSTRUCTIONS PER CYCLE / RELEASE





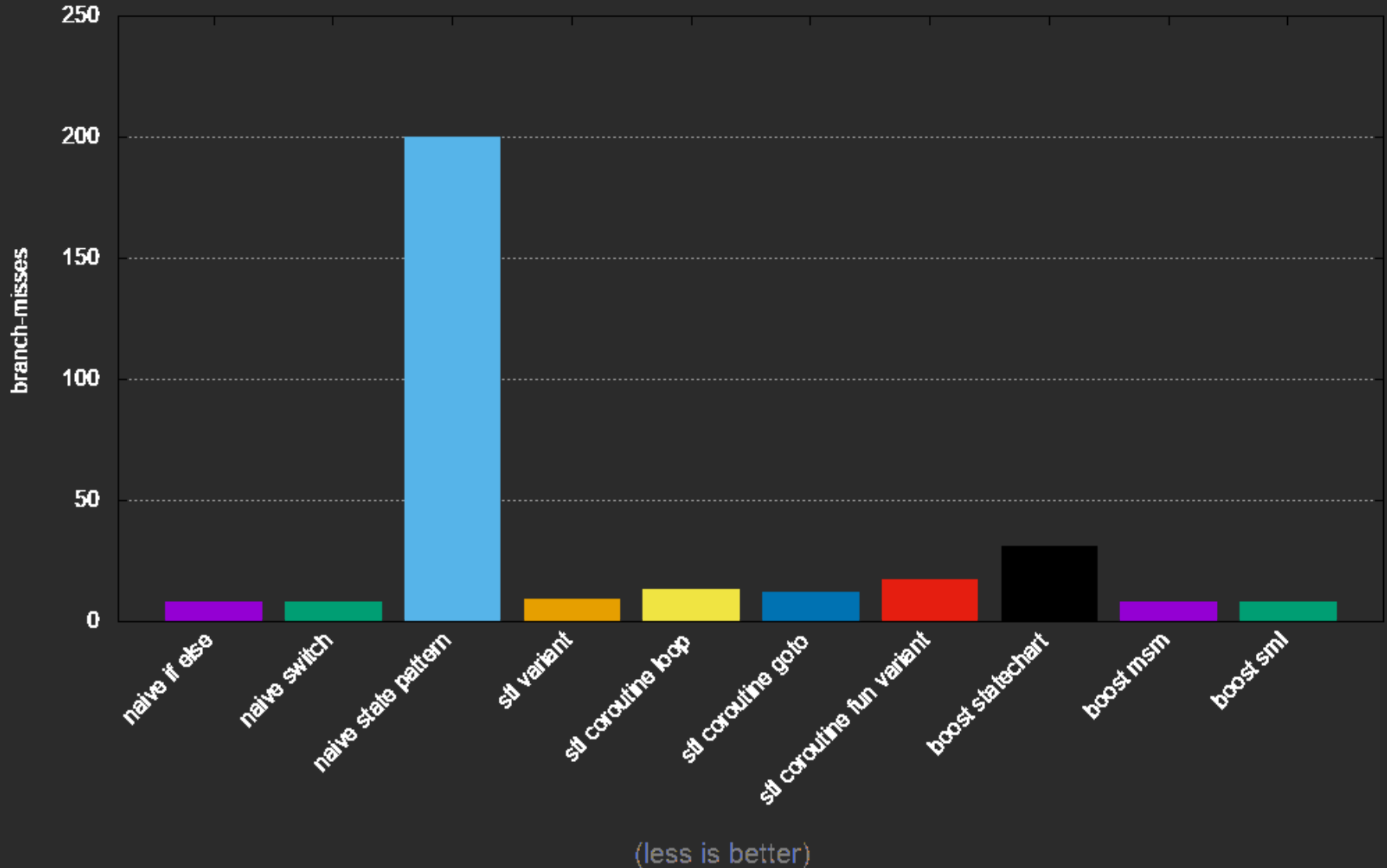


# BENCHMARKS - RUN-TIME PERFORMANCE - BRANCHES / RELEASE



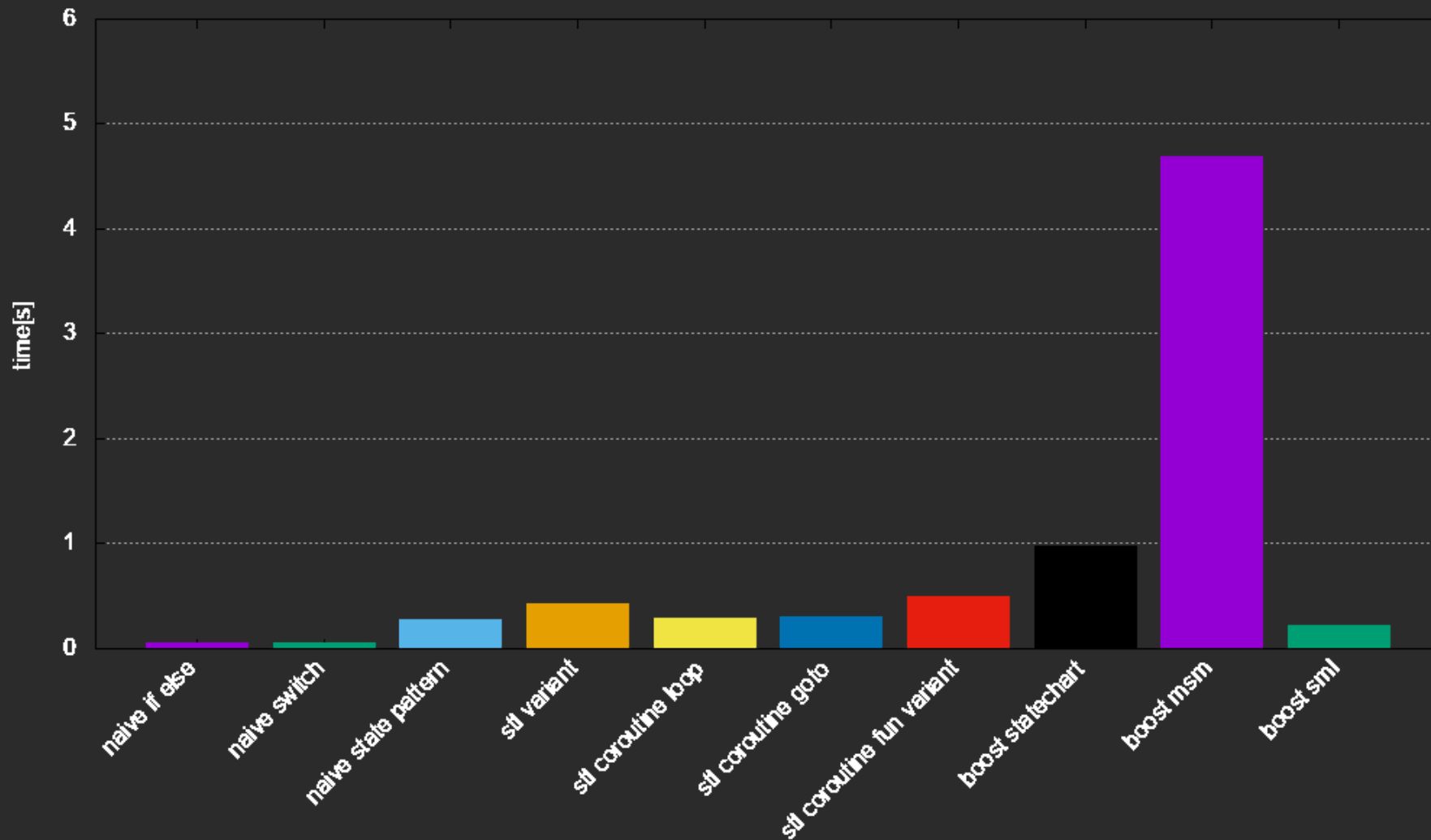


# BENCHMARKS - RUN-TIME PERFORMANCE - BRANCH MISSES / RELEASE





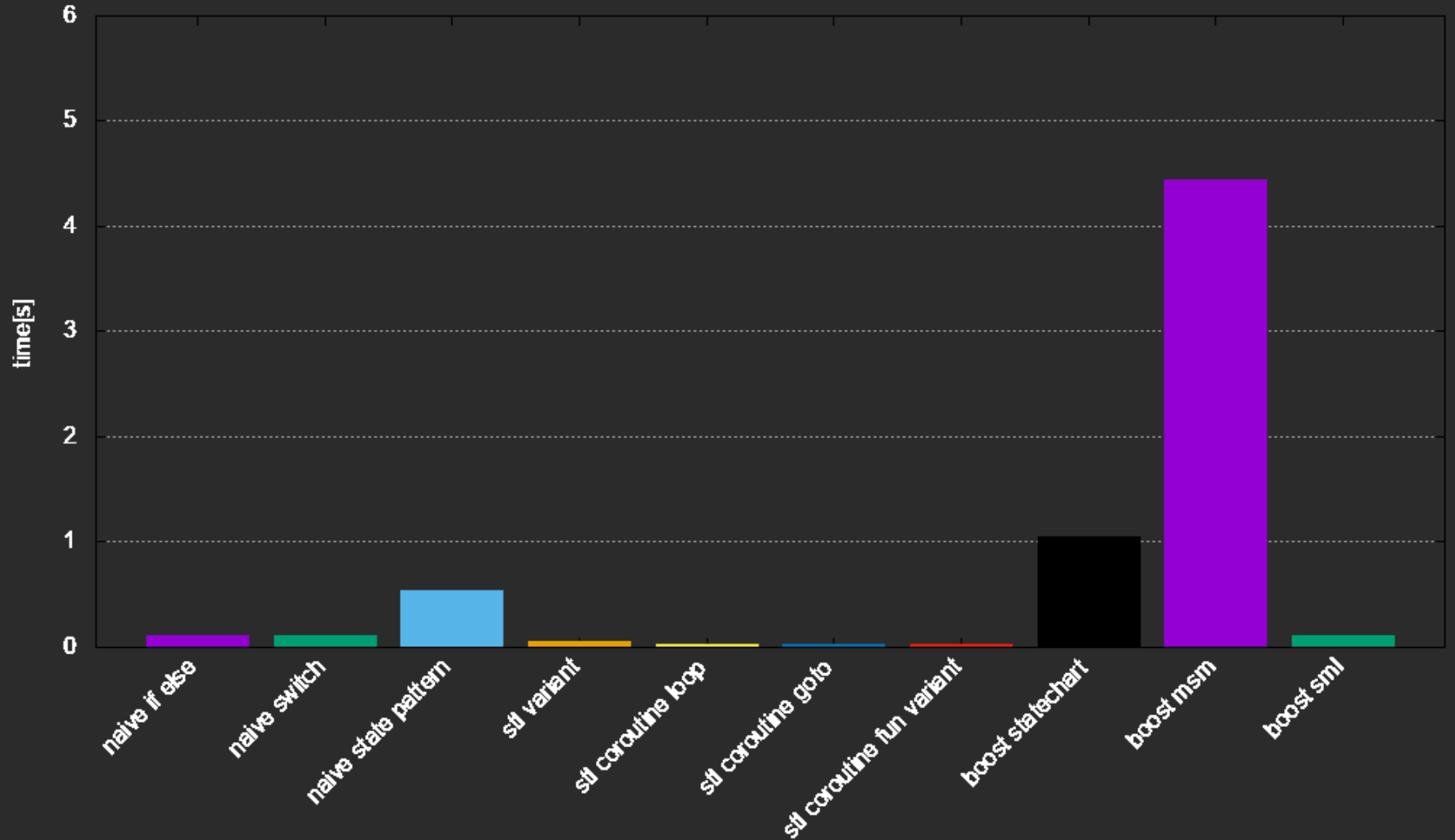
# BENCHMARKS - COMPILATION TIME / DEBUG



(less is better)



# BENCHMARKS - COMPILATION TIME / RELEASE

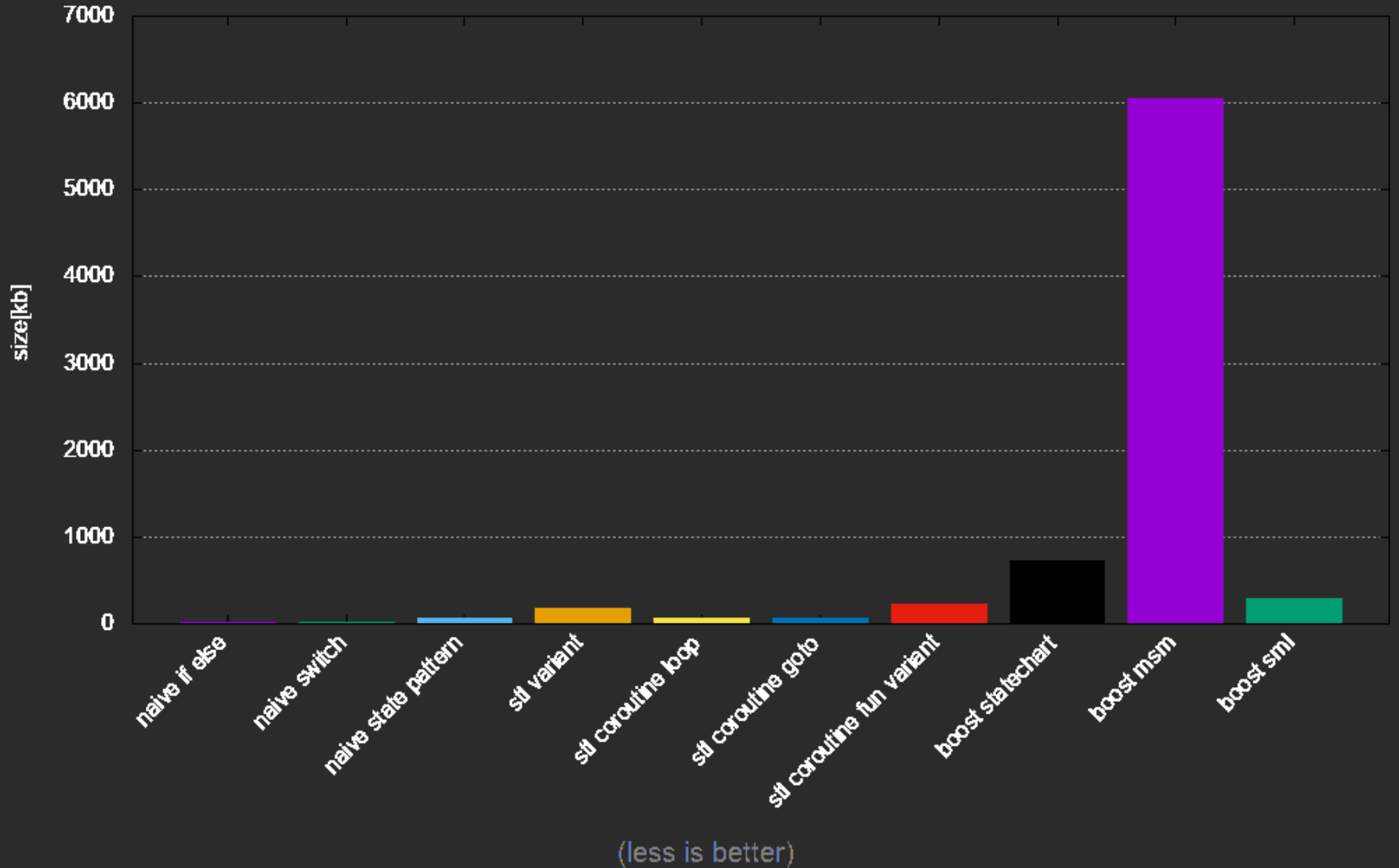


(less is better)



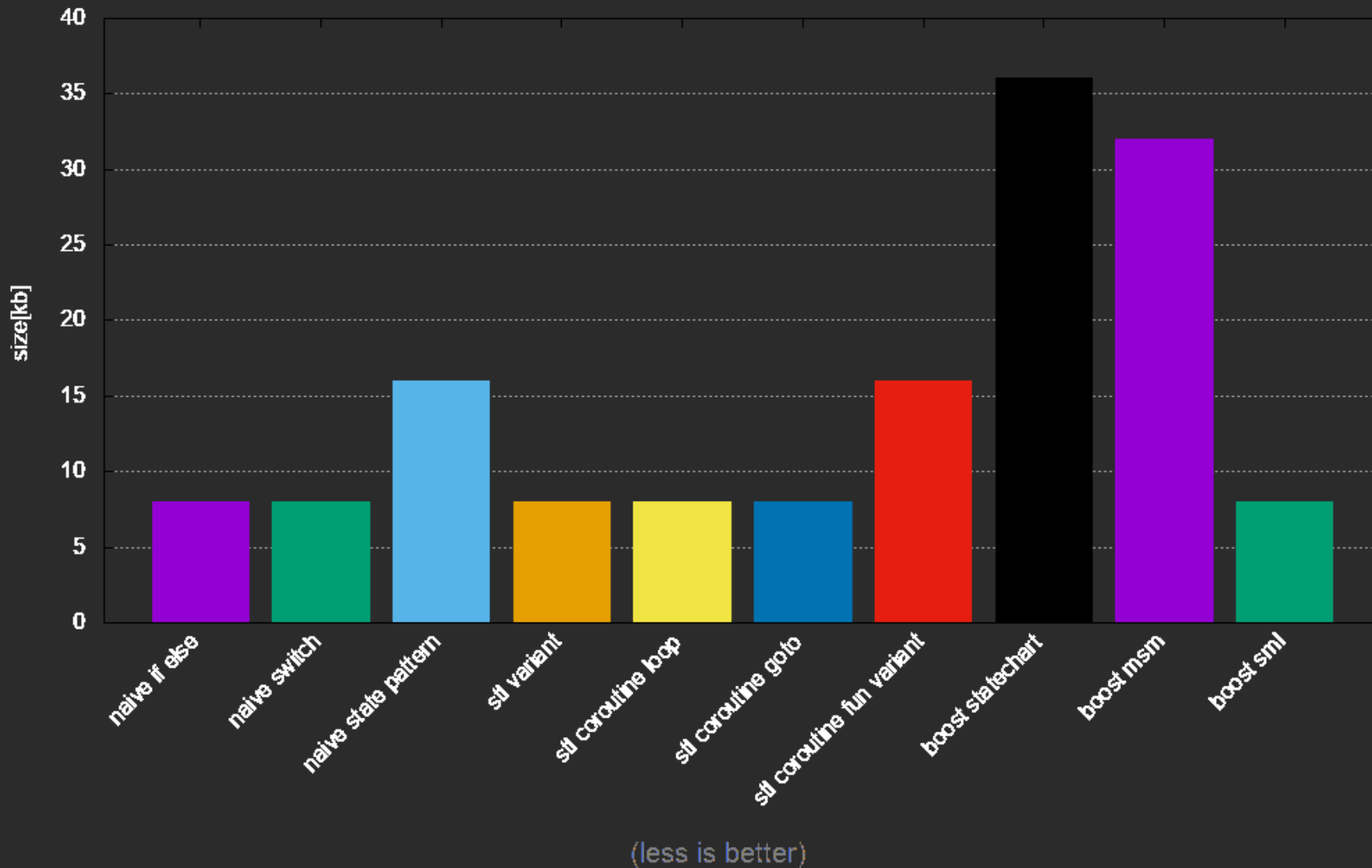


# BENCHMARKS - EXECUTABLE SIZE / DEBUG





# BENCHMARKS - EXECUTABLE SIZE / RELEASE





# LET'S EMBRACE ZERO-COST STATE MACHINE LIBRARIES!

---

Benchmarks <https://github.com/boost-experimental/sml/tree/master/benchmark/connection>

---

Slides <http://boost-experimental.github.io/sml/cppcon-2018>

-

[KRIS@JUSIAK.NET](mailto:KRIS@JUSIAK.NET) | [@KRISJUSIAK](https://twitter.com/KRISJUSIAK) | [LINKEDIN.COM/IN/KRIS-JUSIAK](https://www.linkedin.com/in/kris-jusiak)

# LET'S EMBRACE ZERO-COST STATE MACHINE LIBRARIES!

---

Benchmarks <https://github.com/boost-experimental/sml/tree/master/benchmark/connection>

---

Slides <http://boost-experimental.github.io/sml/cppcon-2018>

-

[KRIS@JUSIAK.NET](mailto:KRIS@JUSIAK.NET) | [@KRISJUSIAK](https://twitter.com/KRISJUSIAK) | [LINKEDIN.COM/IN/KRIS-JUSIAK](https://www.linkedin.com/in/kris-jusiak)

# LET'S EMBRACE ZERO-COST STATE MACHINE LIBRARIES!

---

Benchmarks <https://github.com/boost-experimental/sml/tree/master/benchmark/connection>

---

Slides <http://boost-experimental.github.io/sml/cppcon-2018>

-

[KRIS@JUSIAK.NET](mailto:KRIS@JUSIAK.NET) | [@KRISJUSIAK](https://twitter.com/KRISJUSIAK) | [LINKEDIN.COM/IN/KRIS-JUSIAK](https://www.linkedin.com/in/kris-jusiak)