

# An Executable Formal Semantics of C++

Jan Tušil

FI MU

1.2.2017

# Osnova

1 Kontext

2 Cíle práce

3 Výsledky & co dál?

4 Bonusy

# Osnova

1 Kontext

2 Cíle práce

3 Výsledky & co dál?

4 Bonusy

# Jazyk C++



# Jazyk C++



# Komplexita C/C++

<sup>1</sup><https://runtimeverification.com/blog/?p=200>

# Komplexita C/C++

- nástroje: clang-tidy, frama-c, cbmc, divine, symbiotic

---

<sup>1</sup><https://runtimeverification.com/blog/?p=200>

# Komplexita C/C++

- nástroje: clang-tidy, frama-c, cbmc, divine, symbiotic
- „Rozumí“ nástroje C++ správně?

---

<sup>1</sup><https://runtimeverification.com/blog/?p=200>

# Komplexita C/C++

- nástroje: clang-tidy, frama-c, cbmc, divine, symbiotic
- „Rozumí“ nástroje C++ správně?
- SV-COMP: nedefinované programy v balíku korektních <sup>1</sup>

---

<sup>1</sup><https://runtimeverification.com/blog/?p=200>

# Komplexita C/C++

- nástroje: clang-tidy, frama-c, cbmc, divine, symbiotic
- „Rozumí“ nástroje C++ správně?
- SV-COMP: nedefinované programy v balíku korektních <sup>1</sup>
- možný přístup: delegovat porozumění jazyku na překladač (clang/llvm)

---

<sup>1</sup><https://runtimeverification.com/blog/?p=200>

# Komplexita C/C++

- nástroje: clang-tidy, frama-c, cbmc, divine, symbiotic
- „Rozumí“ nástroje C++ správně?
- SV-COMP: nedefinované programy v balíku korektních <sup>1</sup>
- možný přístup: delegovat porozumění jazyku na překladač (clang/llvm)
- takto ale nelze ověřit korektnost programu vzhledem k jazyku

---

<sup>1</sup><https://runtimeverification.com/blog/?p=200>

# Komplexita C/C++

- nástroje: clang-tidy, frama-c, cbmc, divine, symbiotic
- „Rozumí“ nástroje C++ správně?
- SV-COMP: nedefinované programy v balíku korektních <sup>1</sup>
- možný přístup: delegovat porozumění jazyku na překladač (clang/llvm)
- takto ale nelze ověřit korektnost programu vzhledem k jazyku
- jak verifikovat šablony?

---

<sup>1</sup><https://runtimeverification.com/blog/?p=200>

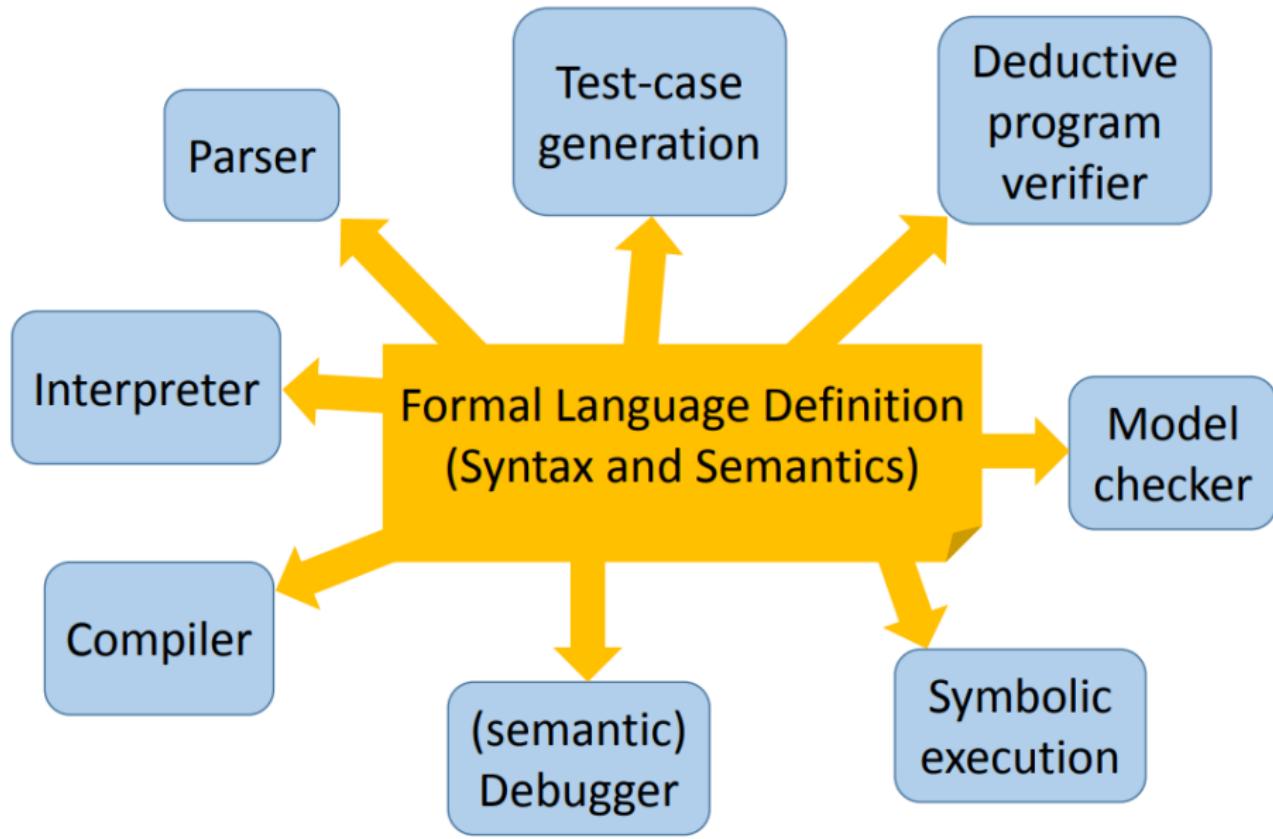
# Komplexita C/C++

- nástroje: clang-tidy, frama-c, cbmc, divine, symbiotic
- „Rozumí“ nástroje C++ správně?
- SV-COMP: nedefinované programy v balíku korektních <sup>1</sup>
- možný přístup: delegovat porozumění jazyku na překladač (clang/llvm)
- takto ale nelze ověřit korektnost programu vzhledem k jazyku
- jak verifikovat šablony?
- jiný přístup: samostatná formální sémantika

---

<sup>1</sup><https://runtimeverification.com/blog/?p=200>

# K Framework



- Již formalizováno: Python, JavaScript, C

- Již formalizováno: Python, JavaScript, C
- využití C: RV-Match



- Již formalizováno: Python, JavaScript, C
- využití C: RV-Match



- experimentálně: C rozšiřováno o podporu C++

# Osnova

1 Kontext

2 Cíle práce

3 Výsledky & co dál?

4 Bonusy

## Výčtové typy (enum)

```
enum E0 { A = 5, B = A + 3 };
enum E1 : char;
enum class E2 : unsigned int;
enum class E3;
enum E5 { A, B } __attribute__((packed));
std::underlying_type_t<E5> x = 5;
```

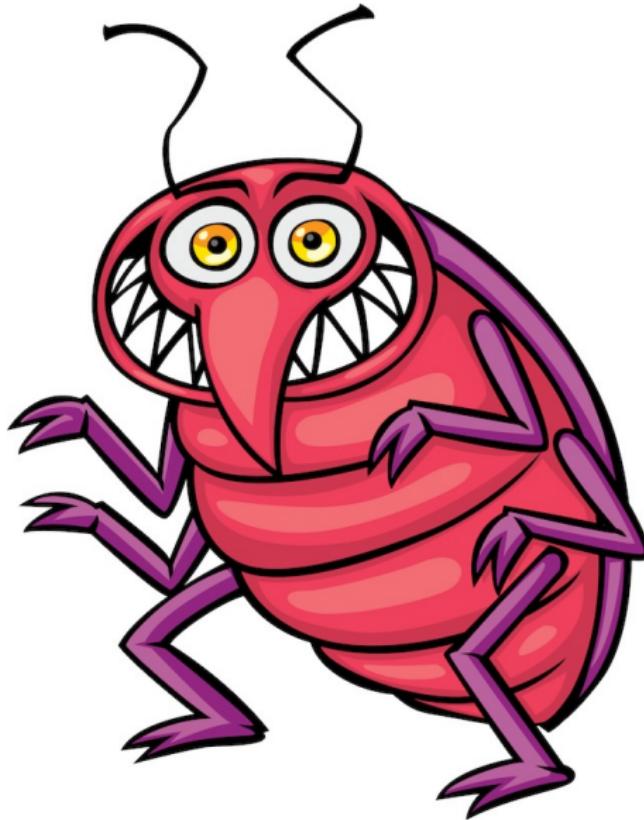
## Vykonávání kódu v době překladu (constexpr)

```
constexpr int fact(int n) {
    return n <= 1 ? 1 : n * fact(n - 1);
}
int main() {
    constexpr int f_4 = fact(4);
    enum class Facts {
        F_4 = f_4,
        F_5 = fact(5)
    };
    assert(int(Facts::F_4) == 24);
    assert(int(Facts::F_5) == 120);
}
```

# Nulová inicializace tříd

```
struct S {  
    virtual ~S() = default;  
    int x;  
};  
  
S s1;  
S s2{};  
assert(s1.x == 0); // ok  
assert(s2.x == 0); // fail
```

# Oprava stávajících chyb v projektu



# Osnova

1 Kontext

2 Cíle práce

3 Výsledky & co dál?

4 Bonusy

# Výsledky

# Výsledky

- enum - vše hotovo, většina je v upstream

# Výsledky

- enum - vše hotovo, většina je v upstream
- constexpr - elegantní proof of concept

# Výsledky

- enum - vše hotovo, většina je v upstream
- constexpr - elegantní proof of concept
- nulová inicializace - hotovo

# Výsledky

- enum - vše hotovo, většina je v upstream
- constexpr - elegantní proof of concept
- nulová inicializace - hotovo
- opravy chyb - některé přijaty do upstreamu

# Výsledky

- enum - vše hotovo, většina je v upstream
- constexpr - elegantní proof of concept
- nulová inicializace - hotovo
- opravy chyb - některé přijaty do upstreamu
- nalezené chyby v GCC, Clang, standardu C++

# Co dál?

# Co dál?

- Vše hotové do upstreamu.

# Co dál?

- Vše hotové do upstreamu.
- Dokončit constexpr.

# Co dál?

- Vše hotové do upstreamu.
- Dokončit `constexpr`.
- Šablony, vlákna, bitová pole, fold expressions, ...

# Co dál?

- Vše hotové do upstreamu.
- Dokončit `constexpr`.
- Šablony, vlákna, bitová pole, fold expressions, ...
- Jak řešit vývoj jazyka?

# Co dál?

- Vše hotové do upstreamu.
- Dokončit `constexpr`.
- Šablony, vlákna, bitová pole, fold expressions, ...
- Jak řešit vývoj jazyka?
- Spolupráce s RuntimeVerification inc.

# Co dál?

- Vše hotové do upstreamu.
- Dokončit `constexpr`.
- Šablony, vlákna, bitová pole, fold expressions, ...
- Jak řešit vývoj jazyka?
- Spolupráce s RuntimeVerification inc.
- Kontrakty.

# Ochutnávka

```
template <typename It>
bool is_sorted(It begin, It end);

/*
 * \pre valid_range(begin, end);
 * \post is_sorted(begin, end);
 */
template <typename It>
void sort(It begin, It end);
```

# Kontrakty

# Kontrakty

- Jak jim dát sémantiku?

# Kontrakty

- Jak jim dát sémantiku?
- Volání běžných funkcí v pre/postcondition.

# Kontrakty

- Jak jim dát sémantiku?
- Volání běžných funkcí v pre/postcondition.
- A co když mají efekty? A co když ty jsou ohrazené?

- Jak jim dát sémantiku?
- Volání běžných funkcí v pre/postcondition.
- A co když mají efekty? A co když ty jsou ohrazené?
- Jak je verifikovat?

# Kontrakty

- Jak jim dát sémantiku?
- Volání běžných funkcí v pre/postcondition.
- A co když mají efekty? A co když ty jsou ohrazené?
- Jak je verifikovat?
- Jak využít existující abstrakce? A vzory?

## Zdroje

- Mapa C++ <https://fearlesscoder.blogspot.cz/2017/02/the-c17-lands.html> K diagram [http://fsl.cs.illinois.edu/index.php/From\\_Rewriting\\_Logic,\\_to\\_Programming\\_Language\\_Semantics,\\_to\\_Program\\_Verification](http://fsl.cs.illinois.edu/index.php/From_Rewriting_Logic,_to_Programming_Language_Semantics,_to_Program_Verification)
- RV-Match logo <https://runtimeverification.com/match/>
- Brouk <http://cliparting.com/free-bug-clipart-30526/>

# Osnova

1 Kontext

2 Cíle práce

3 Výsledky & co dál?

4 Bonusy

# GCC bug

```
enum E : int;  
enum ::E : int{};
```

# Clang bug

```
namespace N { enum E : int; }
enum N::E : int {A,B};
namespace N {
    int x = int(::N::B); // 1
}
int y = int(::A); // 2
int z = int(A); // 3
```

# C++ standard bug

```
enum class E {
    A,
    B = E::A // 1
};
```

Jaký typ má `E::A v (1)`?