>>> Undefined Behavior
>>> A Theoretical Fight Against an Omniscient C++ Compiler

Tiago Royer

March 27, 2024

`https://cs.uchicago.edu/~royer/theory-lunch.pdf`

>>> C and C++ are Standardized Languages

* Current C++ standard: C++20
  ISO/IEC 14882:2020, `https://www.iso.org/standard/79358.html`
* Current C++ draft: C++23
  `https://open-std.org/JTC1/SC22/WG21/docs/papers/2023/n4950.pdf`
* Current C standard: C17
  ISO/IEC 9899:2018, `https://www.iso.org/standard/74528.html`
* Current C draft: C23
  `https://open-std.org/jtc1/sc22/wg14/www/docs/n3096.pdf`

>>> C and C++ are Standardized Languages

- Current C++ standard: C++20
  ISO/IEC 14882:2020, https://www.iso.org/standard/79358.html
- Current C++ draft: C++23
  https://open-std.org/JTC1/SC22/WG21/docs/papers/2023/n4950.pdf
- Current C standard: C17
  ISO/IEC 9899:2018, https://www.iso.org/standard/74528.html
- Current C draft: C23
  https://open-std.org/jtc1/sc22/wg14/www/docs/n3096.pdf

Note: I will blur the distinction between C and C++ here

>>> Undefined Behavior

The C++ Standard defines a set of “axioms”,
and the behavior of any program satisfying these axioms.
Violating the axioms is **undefined behavior**.

>>> Undefined Behavior

The C++ Standard defines a set of “axioms”,
and the behavior of any program satisfying these axioms.
Violating the axioms is **undefined behavior**.

“As-if” rule

- “If your program satisfies the axioms of the standard,
  then the compiler can return any semantically equivalent binary.”

>>> No bounds check

```c
int array[128];
int foo;
array[128] = 42; // Undefined behavior
```

>>> Null pointer dereferencing

```c
int *p = NULL;
*p = 0; // Undefined behavior
```

>>> Integer overflow

```c
// Unoptimized
int f(int x) {
    return (x * 2) / 2;
}

// Optimized
int f(int x) {
    return x;
}

// Cannot be optimized
unsigned int f(unsigned int x) {
    return (x * 2) / 2;
}
```

>>> Integer overflow

```
bool mainGtU(unsigned int i1, unsigned int i2, char *block) {
    char c1, c2;

    /* 1 */
    c1 = block[i1]; c2 = block[i2];
    if(c1 != c2) return (c1 > c2);
    i1++; i2++;

    /* 2 */
    c1 = block[i1]; c2 = block[i2];
    if(c1 != c2) return (c1 > c2);
    i1++; i2++;

    // ...
}
```

There are three things all wise programmers fear:
C's corner cases,
a hardware platform with no documentation,
and the anger of an optimizing compiler.
- JF Bastien

>>> Erasing entire sections of the code

```
// QString did not originally provide null termination

QString inputStr = "class std::vector<int>";
QString result;

for (int index = 0; index < inputStr.size(); ++index) {
    if (inputStr[index+1] == ':' && inputStr[index+2] == ':') {
        index += 2;
        result = input.mid(index);    // expected "vector<int>"
    }
}
```

>>> Disproving Goldbach's Conjecture

```
bool is_prime(int i) { /* Simple O(√n) algorithm */ }

int main() {
    bool counterexample_found = false;
    for(int n = 4; !counterexample_found; n+=2) {
        counterexample_found = true;
        for(int i = 2; i < n; i++)
            if(is_prime(i) && is_prime(n-i))
                counterexample_found = false;
    }

    printf("Counterexample found!\n");
    return 0;
}
```

>>> Time travel

```
void f(SomeClass *p) {
    if(p != NULL) p->doSomething();
    // ...
    // some code
    // ...
    p->doSomethingElse();
}

// This gets optimized to
void f(int *p) {
    p->doSomething();
    // ...
    // some code
    // ...
    p->doSomethingElse();
}
```

>>> Undefined Behavior
>>> A Theoretical Fight Against an Omniscient C++ Compiler

Tiago Royer
March 27, 2024

`https://cs.uchicago.edu/~royer/theory-lunch.pdf`

>>> References

* Beamer theme: DarkConsole
  https://github.com/kmaed/kmbeamer/
* mainGtU example
  https://youtu.be/yG1OZ69H_-o?t=2358
* JF Bastien quote
  https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p1152r0.html
* QString example
  https://youtu.be/NpL9YnxnOqM?t=2773
* C Compilers Disprove Fermat's Last Theorem
  https://blog.regehr.org/archives/140
* A better time travel example
  https://devblogs.microsoft.com/oldnewthing/20140627-00/?p=633