

Examples

```
output(p + a > p + b)
```

```
uintptr_t pi = (uintptr_t) p
```

```
output(pi + a > pi + b)
```

UB Example 1

```
void foo(char* buf, unsigned int len) {
```

```
    /* Wrapping checks */
```

```
    if (buf + len < buf) {
```

```
        return;
```

```
    }
```

```
    /* do something using buf */
```

```
}
```

Scratch

In Current LLVM

- Poison value
- (Bad concept) ~~Undefined value~~

$x = \text{undef}$

->

$x = 10$

$x = \dots$

$y = x + 10$

Compiler Correctness

- Source/IR programs = specifications
- Machine code = implementation
- A programmer write a C program
= She specifies allowed behaviors
- A compiler translates it to machine code
= It gives an implementation satisfying the spec
- A compiler translation is correct
if it preserves or narrows down the spec (ie, behaviors)
- This is called “behavioral refinement”

Example

```
p = malloc(4)
```

In C, `malloc(4)` can allocate a block of size 4 at any free address.

This is a specification.

In machine code, `malloc(4)` will allocate a block at a certain address according to the algorithm of `malloc`.

This is an implementation.