Multiplication and Division in the Range-Based Constraint Manager

Ádám Balogh
adam.balogh@ericsson.com

Range-Based Constraint Manager

— Default in Clang Static Analyzer

Range-Based Constraint Manager

- Default in Clang Static Analyzer
- Good performance: more than 20 times faster than MS Z3 (our measurement)

Range-Based Constraint Manager

- Default in Clang Static Analyzer
- Good performance: more than 20 times faster than MS Z3 (our measurement)
- Limited deduction capabilities: only symbol plus/minus concrete integer compared to another integer

Range-Based Constraint Manager

- Default in Clang Static Analyzer
- Good performance: more than 20 times faster than MS Z3 (our measurement)
- Limited deduction capabilities: only symbol plus/minus concrete integer compared to another integer

Code

```
signed char n = get_number();

assert(i >= 100);

assert(i + 20 <= -120);
```

Ranges

```
n: [-128..127]

n: [100..127] == [-128..127]&[100..127]

n: [108..116] == [100..127]&([-128..-120]-20)
```

The Problem: False Positive

— The result of multiplicative operations is unknown:

true_negative.c

```
int size = 4, n, i;
for (i = 0; i < size - 2; ++i)
  init(&n);
use(n); // no warning
```

false_positive.c

```
int size = 4, n, i;
for (i = 0; i < size / 2; ++i)
  init(&n);
use(n); // warning: n unitialized
```

The Problem: False Positive

— The result of multiplicative operations is unknown:

true_negative.c

```
int size = 4, n, i;
for (i = 0; i < size - 2; ++i)
  init(&n);
use(n); // no warning
```

false_positive.c

```
int size = 4, n, i;
for (i = 0; i < size / 2; ++i)
  init(&n);
use(n); // warning: n unitialized
```

The Problem: False Positive

— The result of multiplicative operations is unknown:

true_negative.c

```
int size = 4, n, i;
for (i = 0; i < size - 2; ++i)
  init(&n);
use(n); // no warning
```

false_positive.c

```
int size = 4, n, i;
for (i = 0; i < size / 2; ++i)
  init(&n);
use(n); // warning: n unitialized
```

— Z3 refutation may help to get rid of these false positives

The Problem: False Negative

— Z3 refutation, does not help to get rid of false negatives

true_positive.c

```
int n = get_number();
assert (n <= 2);
assert (n + 2 >= 4);
1 / (n - 2); // div. by zero
```

false_negative.c

```
int n = get_number();
assert (n <= 2);
assert (n * 2 >= 4);
1 / (n - 2); // no warning
```

The Problem: False Negative

— Z3 refutation, does not help to get rid of false negatives

true_positive.c

```
int n = get_number();
assert (n <= 2);
assert (n + 2 >= 4);
1 / (n - 2); // div. by zero
```

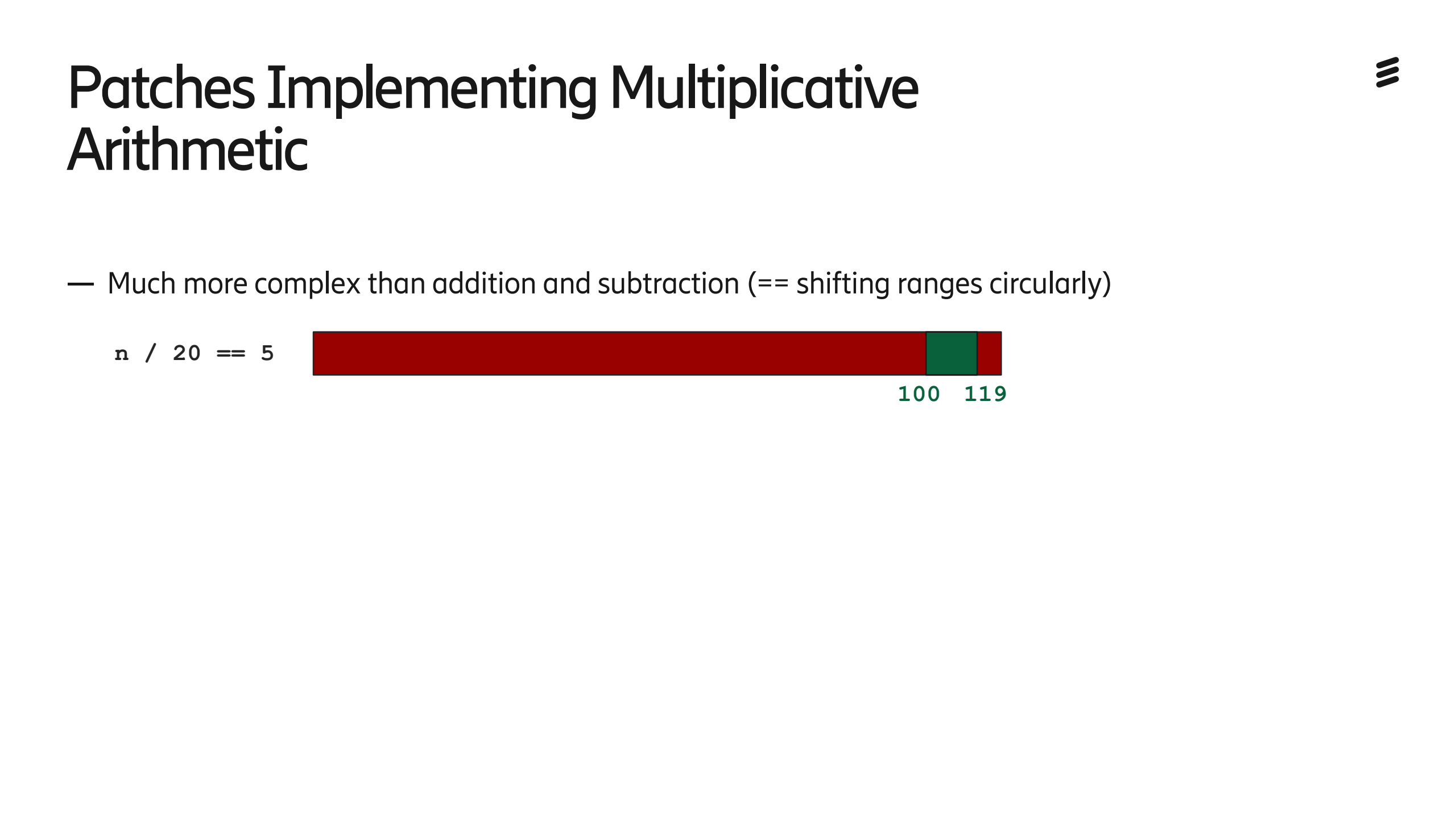
false_negative.c

```
int n = get_number();
assert (n <= 2);
assert (n * 2 >= 4);
1 / (n - 2); // no warning
```

Patches Implementing Multiplicative Arithmetic

— Much more complex than addition and subtraction (== shifting ranges circularly)

Patches Implementing Multiplicative Arithmetic



— Much more complex than addition and subtraction (== shifting ranges circularly)

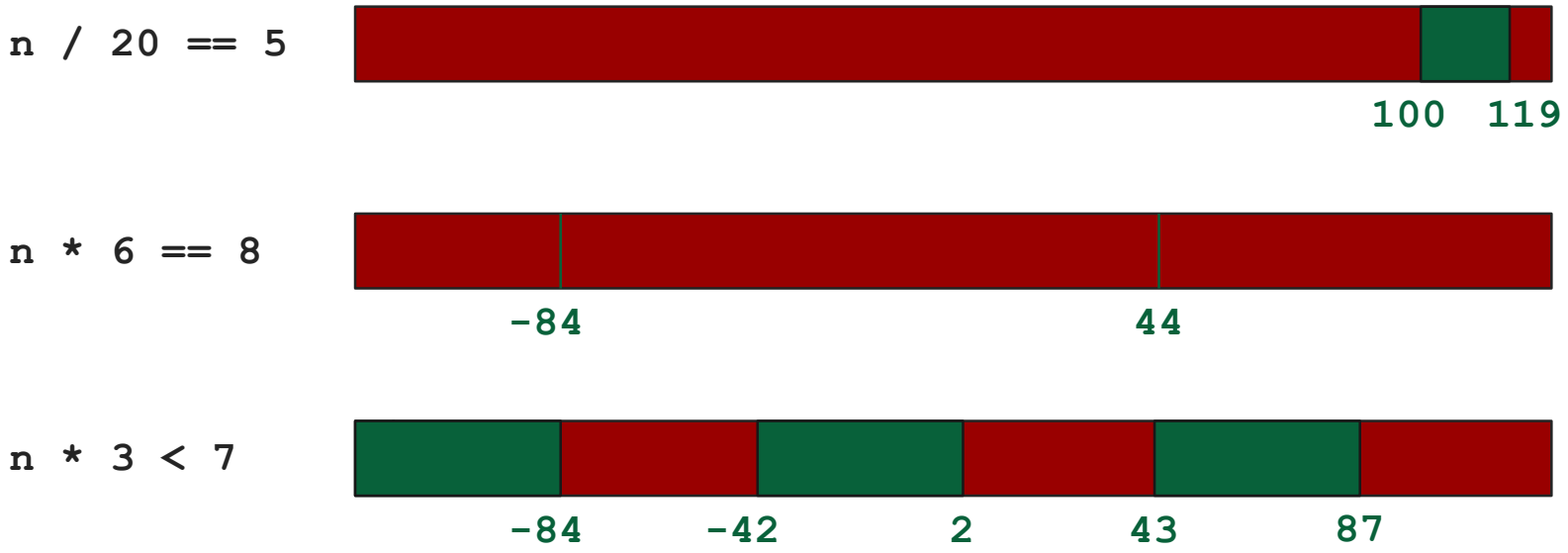
```
n / 20 == 5
```

Patches Implementing Multiplicative Arithmetic

— Much more complex than addition and subtraction (== shifting ranges circularly)

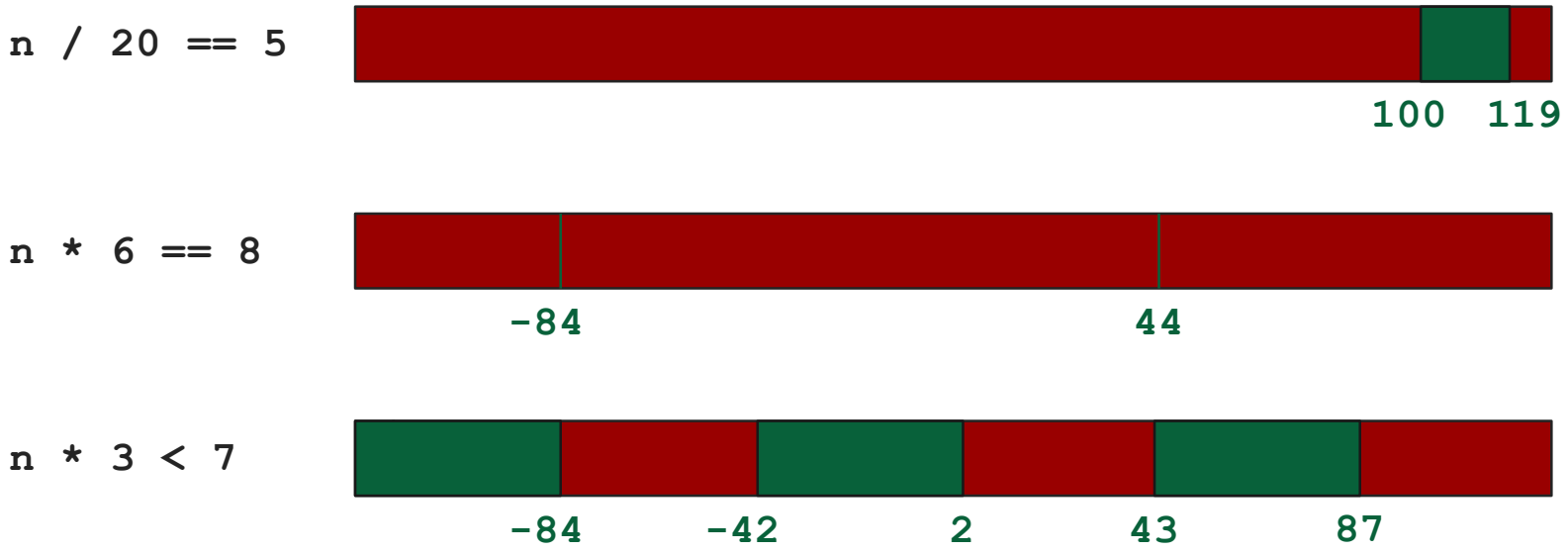
Patches Implementing Multiplicative Arithmetic

— Much more complex than addition and subtraction (== shifting ranges circularly)



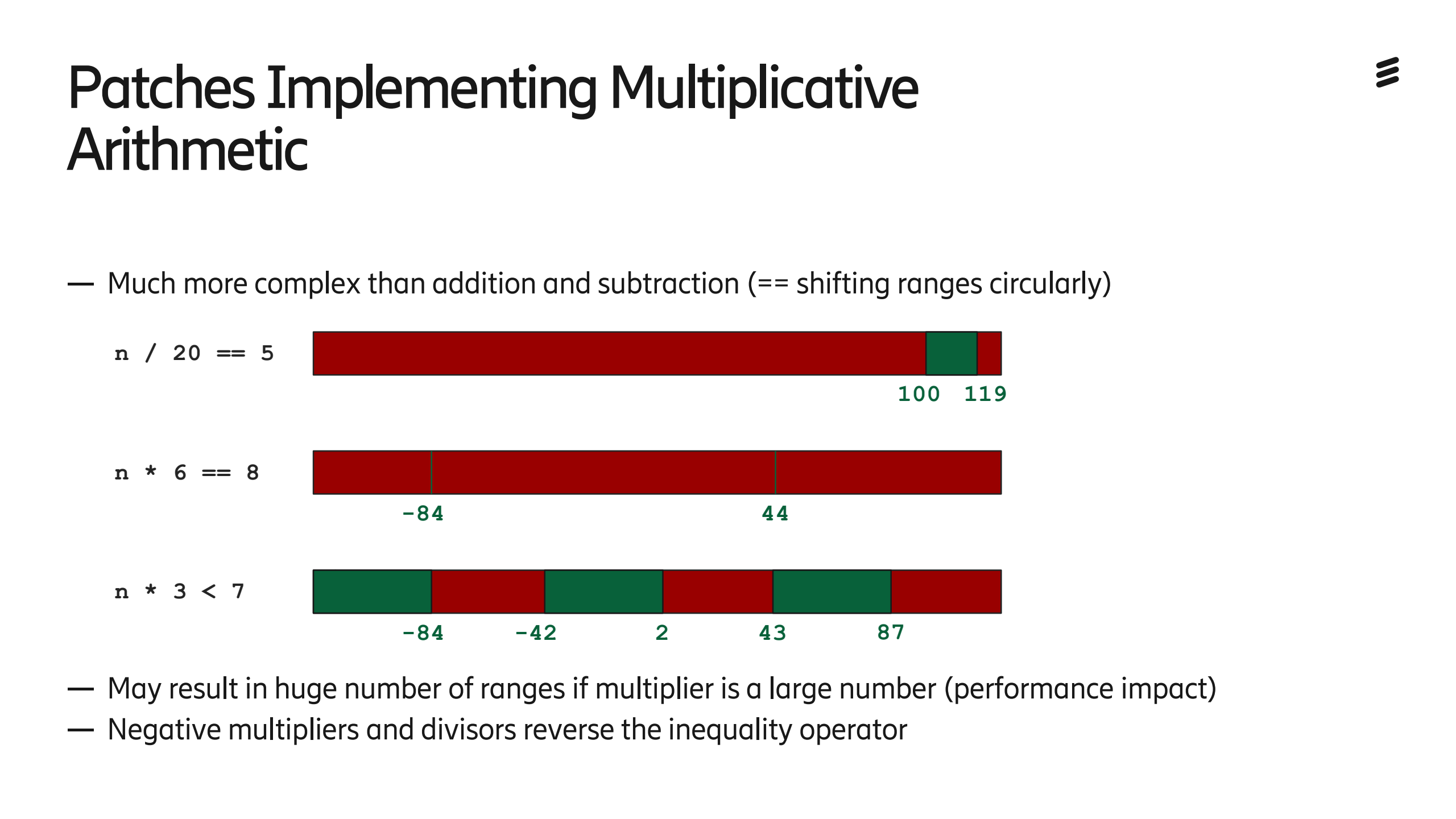
Patches Implementing Multiplicative Arithmetic

— Much more complex than addition and subtraction (== shifting ranges circularly)



— May result in huge number of ranges if multiplier is a large number (performance impact)

Patches Implementing Multiplicative Arithmetic

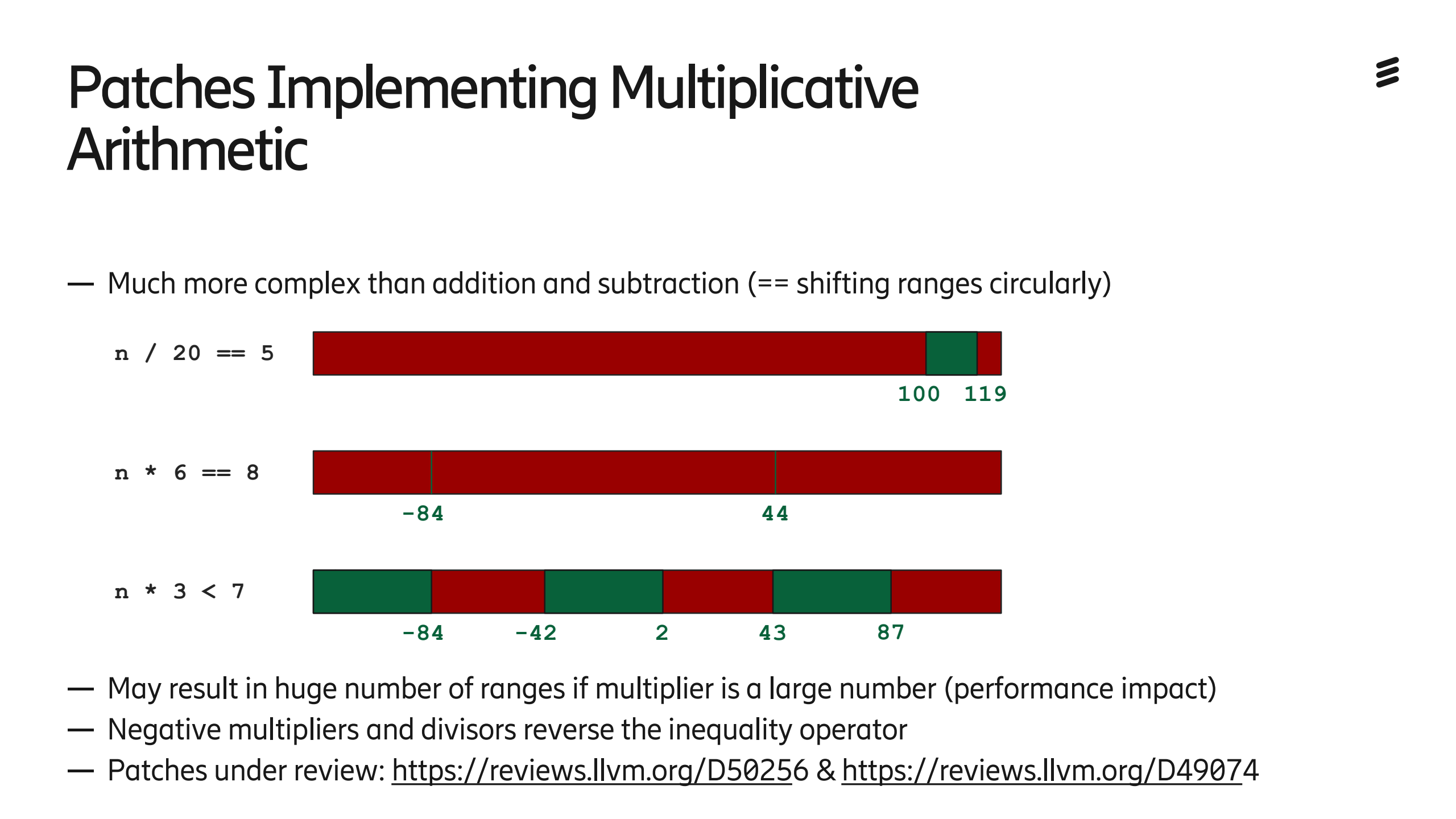


— Much more complex than addition and subtraction (== shifting ranges circularly)

— May result in huge number of ranges if multiplier is a large number (performance impact)

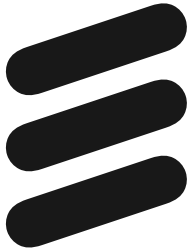
— Negative multipliers and divisors reverse the inequality operator

Patches Implementing Multiplicative Arithmetic



— Much more complex than addition and subtraction (== shifting ranges circularly)

— May result in huge number of ranges if multiplier is a large number (performance impact)
— Negative multipliers and divisors reverse the inequality operator
— Patches under review: https://reviews.llvm.org/D50256 & https://reviews.llvm.org/D49074



Thank You!

adam.balogh@ericsson.com