Statistics Based Checkers in the Clang Static Analyzer

Ádám Balogh
adam.balogh@ericsson.com

The Problem

— Huge legacy code with weak documentation

The Problem

— Huge legacy code with weak documentation

X.h

```
int may_return_return_negative(); // no body available
```

The Problem

— Huge legacy code with weak documentation

X.h

```
int may_return_return_negative(); // no body available
```

— Many calls for **`may_return_negative()`**, return value is checked for negative in e.g. 98% of the calls

The Problem

— Huge legacy code with weak documentation

X.h

```
int may_return_return_negative(); // no body available
```

— Many calls for **`may_return_negative()`**, return value is checked for negative in e.g. 98% of the calls

Y1.c

```
int i = may_return_return_negative();
if (i < 0)
  return;
v[i]; // OK.
```

The Problem

— Huge legacy code with weak documentation

X.h

```
int may_return_return_negative(); // no body available
```

— Many calls for `may_return_negative()`, return value is checked for negative in e.g. 98% of the calls

Y2.c

```
int i = may_return_return_negative();
if (i < 0)
  return;
v[i]; // OK.
```

The Problem

— Huge legacy code with weak documentation

X.h

```
int may_return_return_negative(); // no body available
```

— Many calls for **`may_return_negative()`**, return value is checked for negative in e.g. 98% of the calls

Y3.c

```
int i = may_return_return_negative();
if (i < 0)
  return;
v[i]; // OK.
```

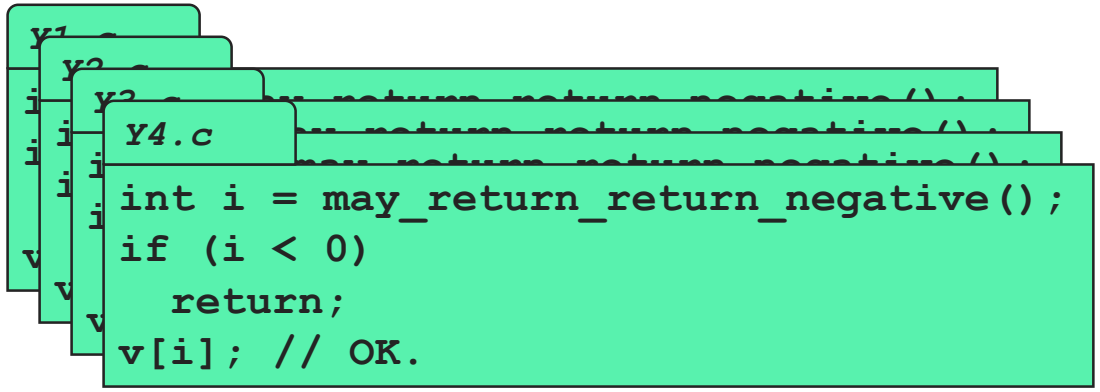
The Problem

— Huge legacy code with weak documentation

X.h

```
int may_return_return_negative(); // no body available
```

— Many calls for `may_return_negative()`, return value is checked for negative in e.g. 98% of the calls



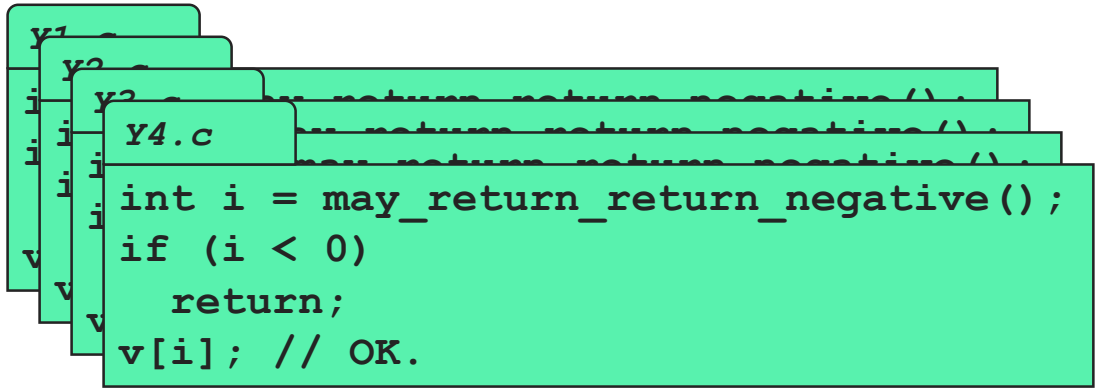
The Problem

— Huge legacy code with weak documentation

X.h

```
int may_return_return_negative(); // no body available
```

— Many calls for **`may_return_negative()`**, return value is checked for negative in e.g. 98% of the calls



Yn.c

```
int i = may_return_return_negative();
v[i]; // error: negative indexing
```

The Problem

— Huge legacy code with weak documentation

X.h

```
int may_return_return_negative(); // no body available
```

— Many calls for `may_return_negative()`, return value is checked for negative in e.g. 98% of the calls

Y4.c

```
int i = may_return_return_negative();
if (i < 0)
  return;
v[i]; // OK.
```

Yn.c

```
int i = may_return_return_negative();
v[i]; // error: negative indexing
```

— Goal: detect the 2% where the negativeness of the return value is not checked

What do we check?

Ignored Return Value

Find calls where the return value is ignored but it should not

Ignored.c

```
fread(data, …);
// data may be garbage
// if read failed
```

What do we check?

Ignored Return Value

Find calls where the return value is ignored but it should not

Ignored.c

```
fread(data, …);
// data may be garbage
// if read failed
```

Special Return Value

Negative Integers

Find calls where the integer return value is not checked for negative but it should

NegRet.c

```
int i = ret_neg();
v[i]; // error
```

What do we check?

Ignored Return Value

Find calls where the return value is ignored but it should not

Ignored.c

```
fread(data, …);
// data may be garbage
// if read failed
```

Special Return Value

Negative Integers

Find calls where the integer return value is not checked for negative but it should

NegRet.c

```
int i = ret_neg();
v[i]; // error
```

Null Pointers

Find calls where the pointer return value is not checked for null pointer but it should

NullRet.c

```
T *t = ret_null();
t->field; // error
```

What do we check?

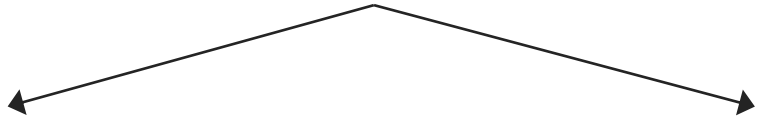
Ignored Return Value

Find calls where the return value is ignored but it should not

Ignored.c

```
fread(data, …);
// data may be garbage
// if read failed
```

Special Return Value



Negative Integers

Find calls where the integer return value is not checked for negative but it should

NegRet.c

```
int i = ret_neg();
v[i]; // error
```

Null Pointers

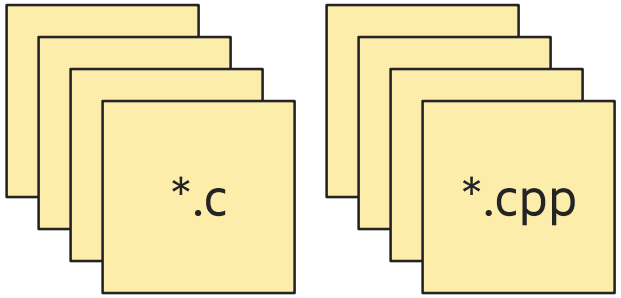
Find calls where the pointer return value is not checked for null pointer but it should

NullRet.c

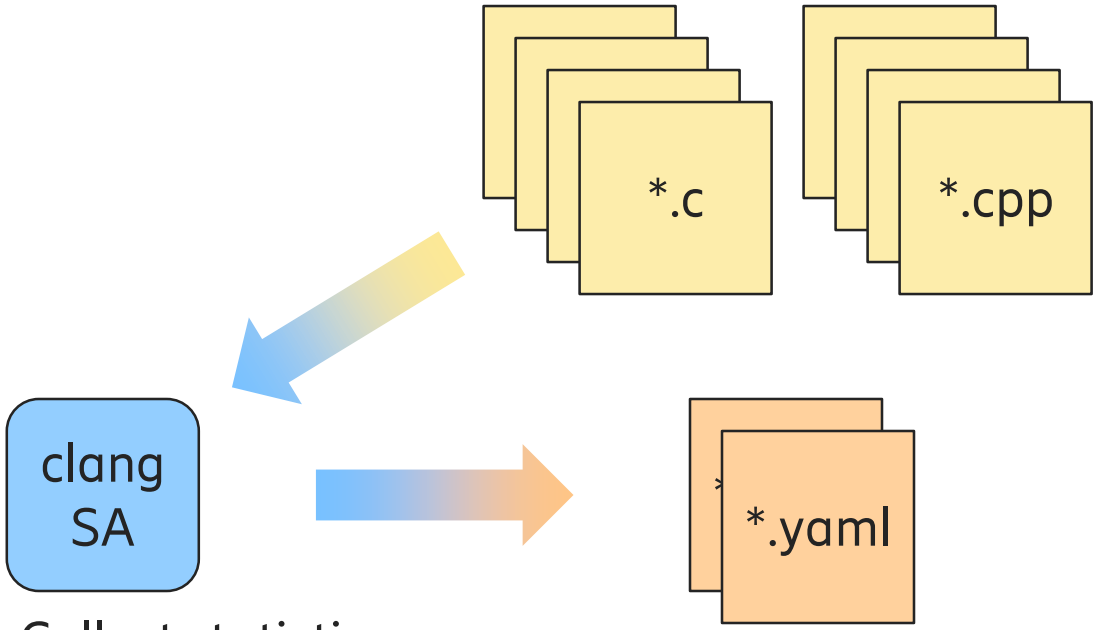
```
T *t = ret_null();
t->field; // error
```

Other “special” values may be added

How does it work?

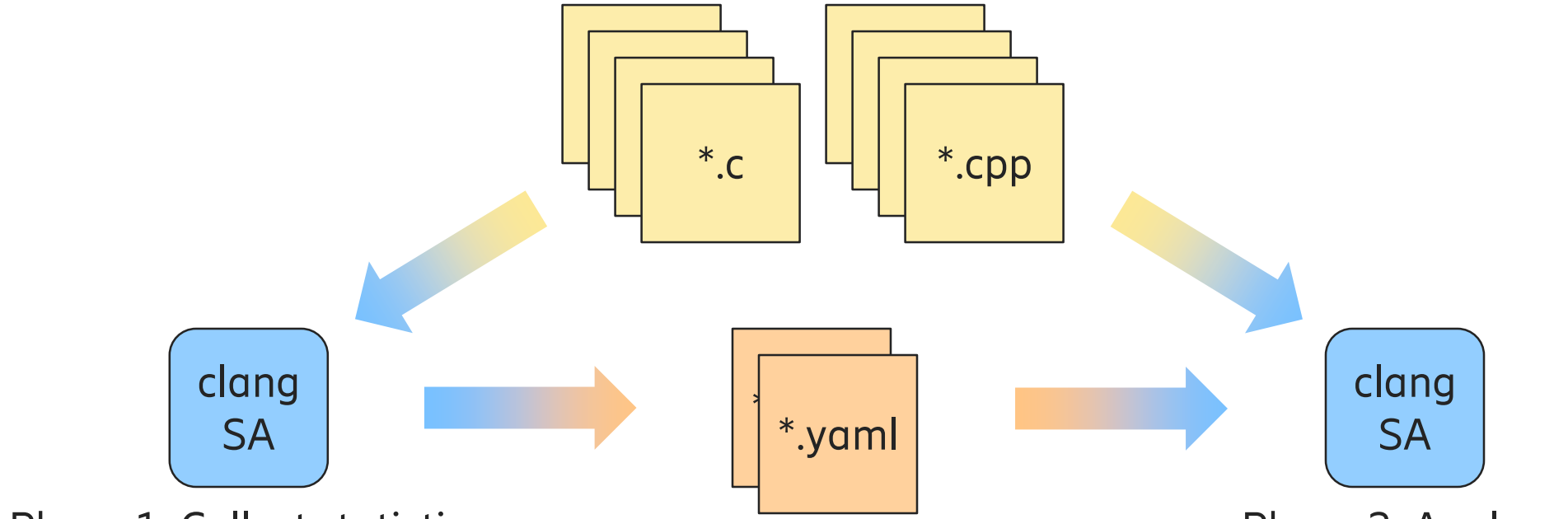


How does it work?



Phase 1: Collect statistics

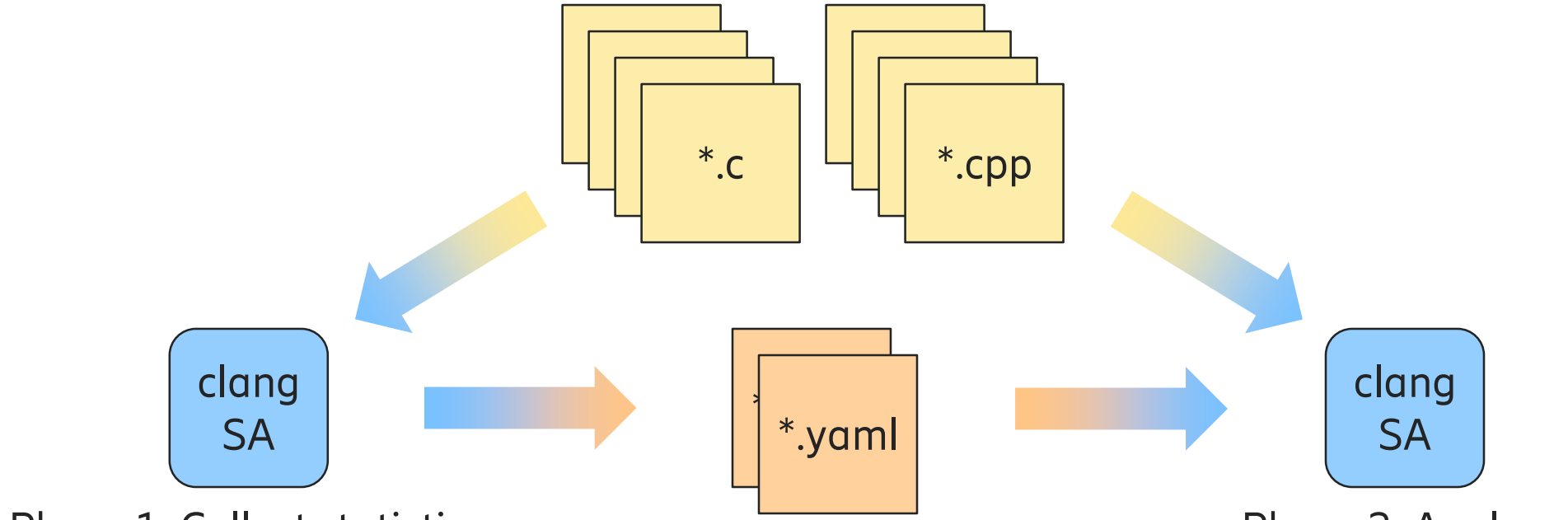
How does it work?



Phase 1: Collect statistics

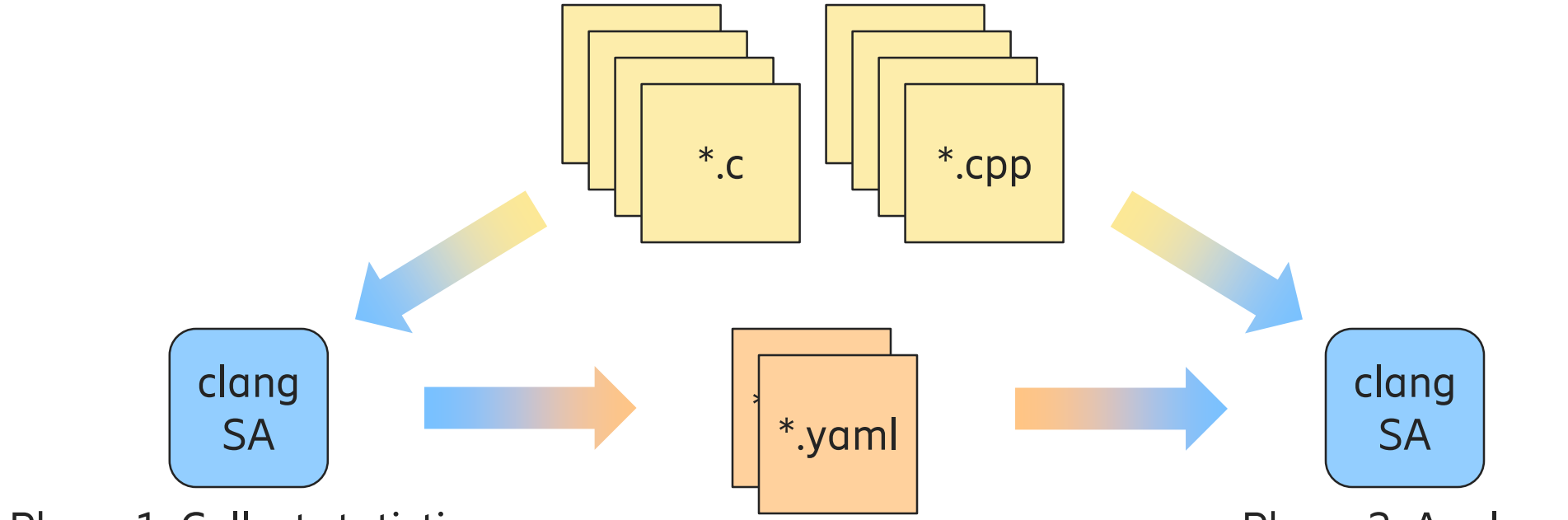
Phase 2: Analyze

How does it work?



— Threshold and minimum required number of calls configurable (default: 85% and 10 calls)

How does it work?



— Threshold and minimum required number of calls configurable (default: 85% and 10 calls)
— **CodeChecker** support exists, open sourcing planned

Checking for Special Return Values

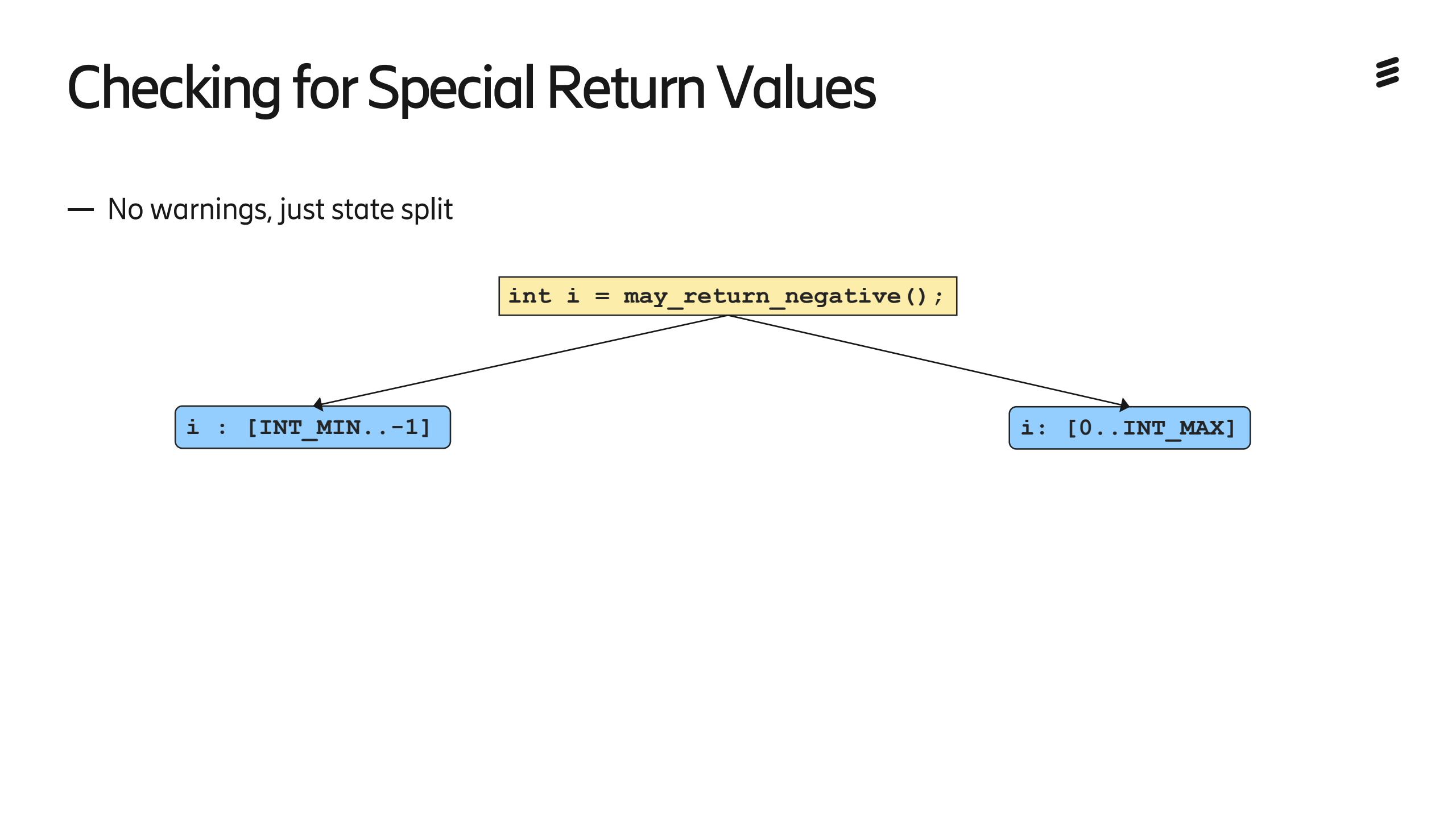
— No warnings, just state split

Checking for Special Return Values

— No warnings, just state split

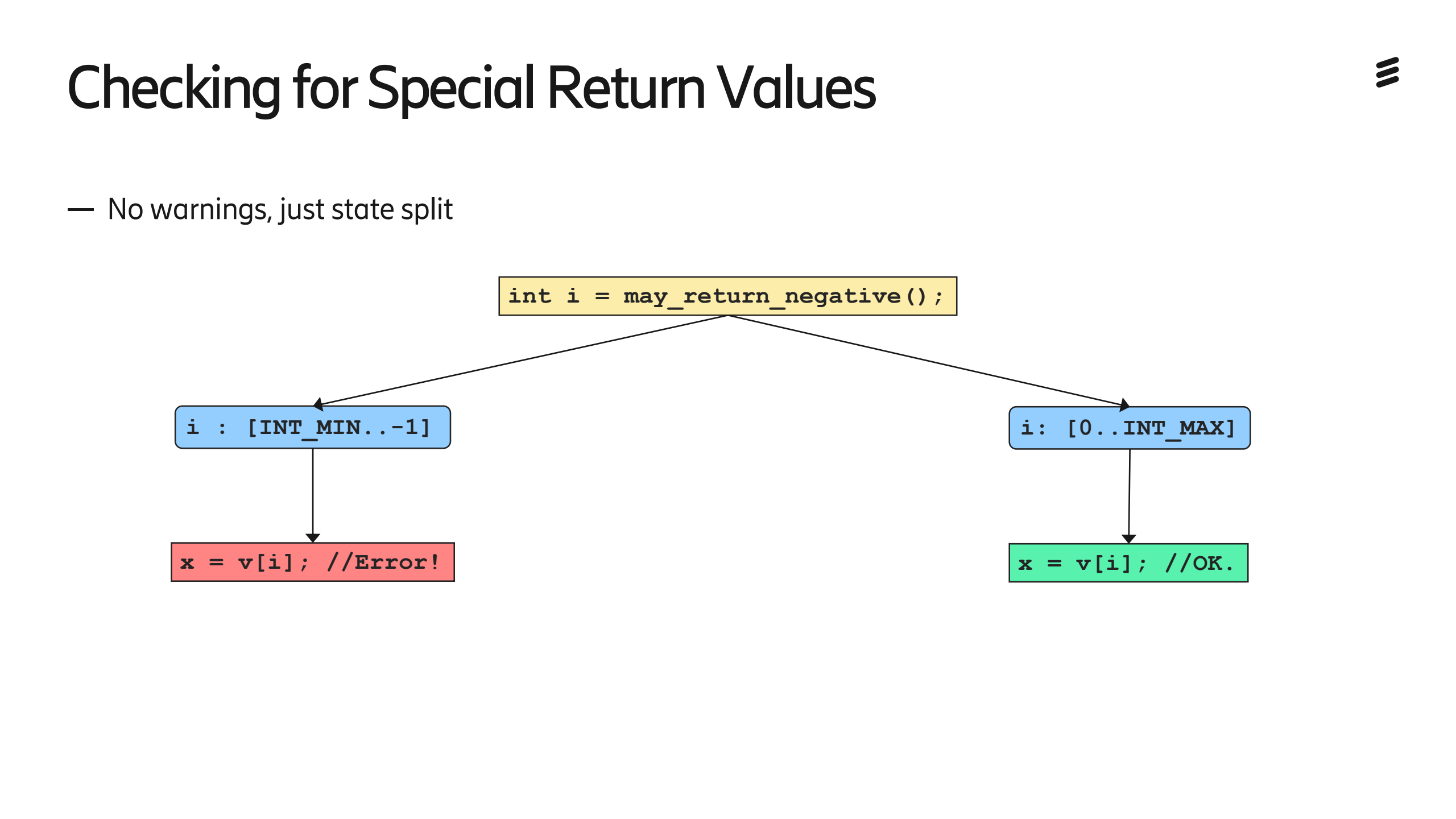
```
int i = may_return_negative();
```

Checking for Special Return Values



— No warnings, just state split

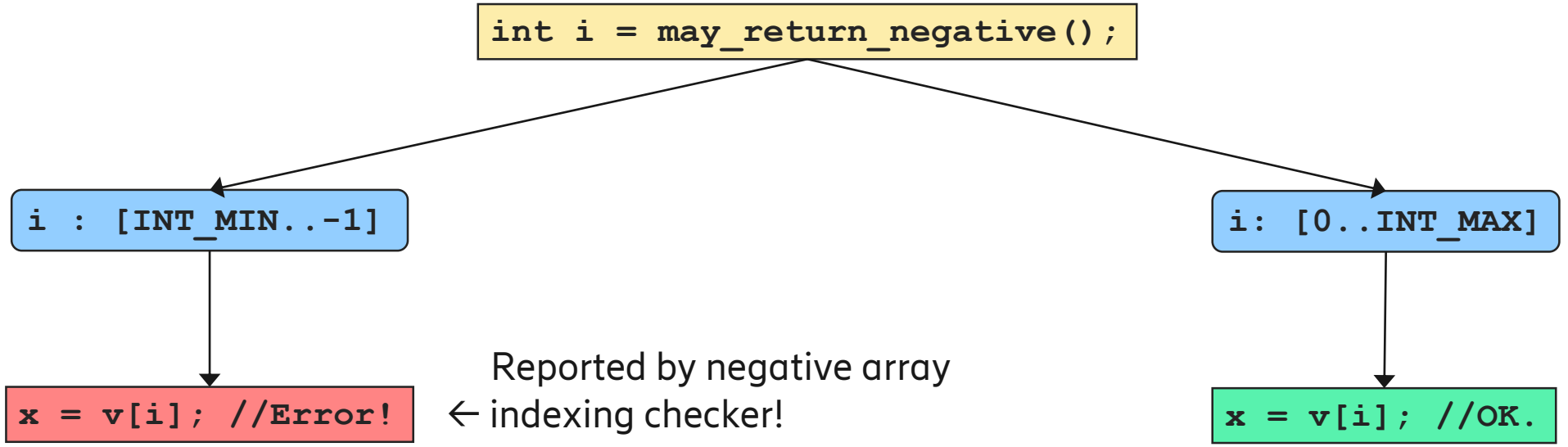
Checking for Special Return Values



— No warnings, just state split

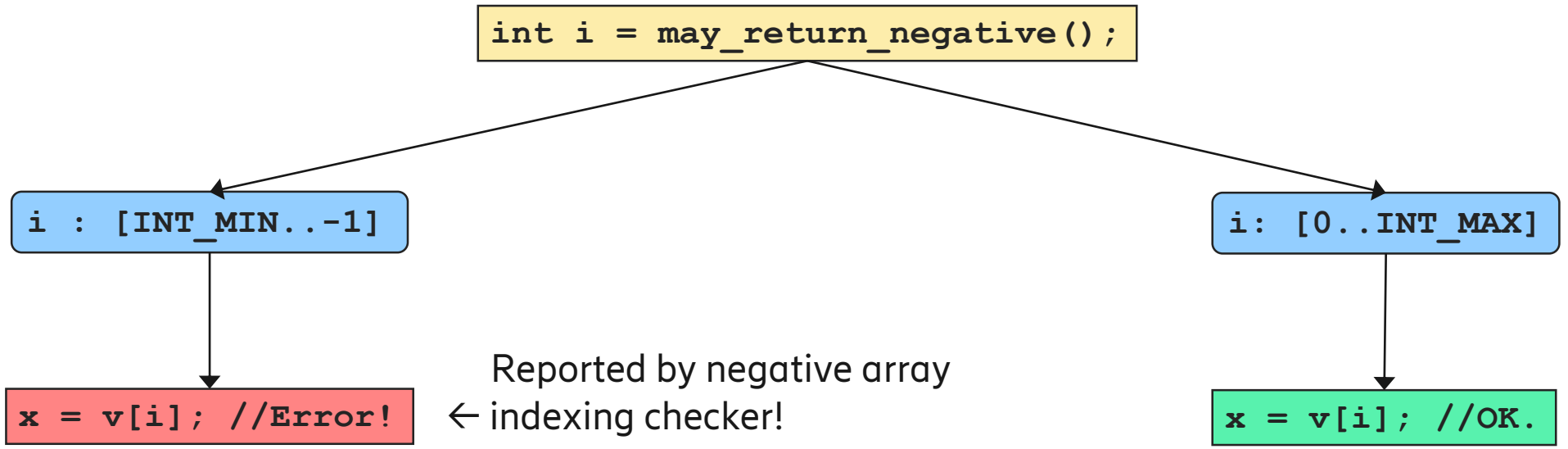
Checking for Special Return Values

— No warnings, just state split



Checking for Special Return Values

— No warnings, just state split



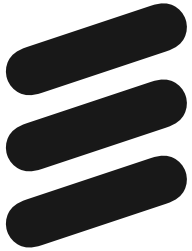
— Performance impact: low because special return value branch terminates quickly
  — Either by early exit or because of error

Future Work

— False Positives: The possible return values often depend from the arguments

Future Work

- False Positives: The possible return values often depend from the arguments
- Solution: Take the parameters also into consideration



Thank You!

adam.balogh@ericsson.com