Finding Bugs using Path-Sensitive Static Analysis

Gábor Horváth
Gabor.Horvath@microsoft.com
@XazaxHun

Welcome to CppCon 2021!

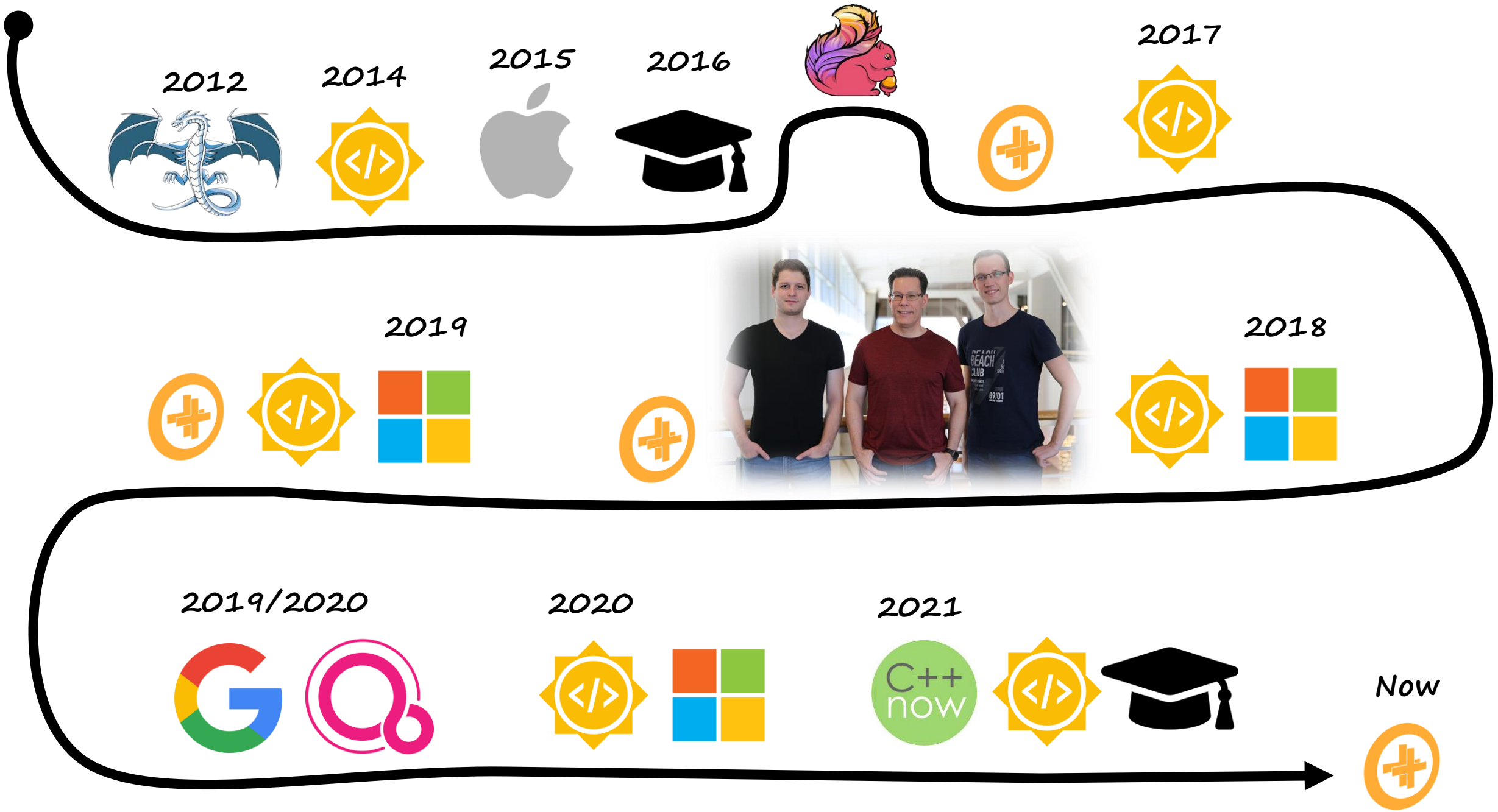
Join #visual_studio channel on CppCon Discord
https://aka.ms/cppcon/discord

- Meet the Microsoft C++ team
- Ask any questions
- Discuss the latest announcements

Take our survey
https://aka.ms/cppcon

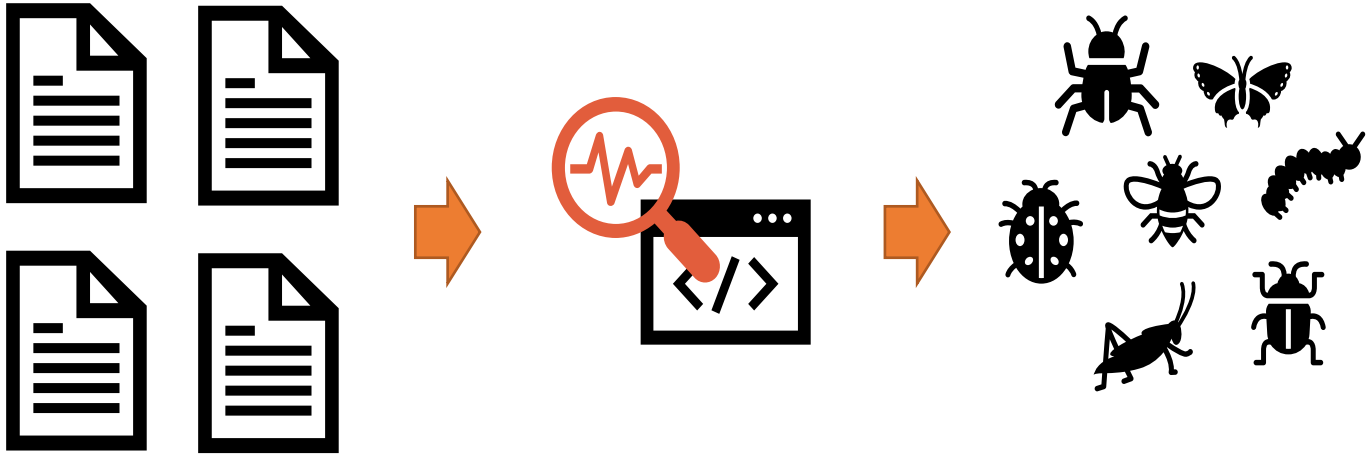
Agenda

- Intro to path-sensitive static analysis
- Path-sensitive checks in MSVC
- A look under the hood
- Upcoming features
- Lessons learned


2012
2014
2015
2016
2017
2018
2019
2019/2020
2020
2021
C++ now
Now

Static Analysis

```
tor<int> v(5);
[0];

/nologo /analyze:only /analyze:plugin E


.cpp(6) : warning C26446: Prefer to us
ubscript operator (bounds.4).
cpp(6) : warning C26816: The pointer
ack.
```



```
int f(int x) {
  if (x == 472349) {
    return 5/(x-472349);
  }
  // ...
}
```

File Edit Selection ··· pathsensitive.cpp - Visu...

pathsensitive.cpp

Drive - Microsoft > Documents > pathsensitive.cpp > path_sensitive(int *

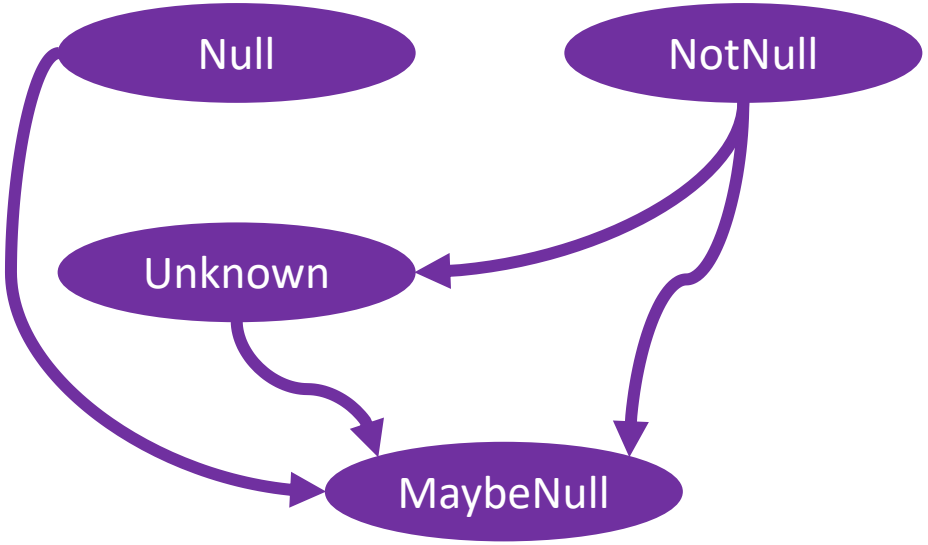
```cpp
void path_sensitive(int *p, bool cond) {
    int state = 0;

    // branch 1
    if (p != nullptr) {
        state = 1;
    }

    // branch 2
    if (cond) {
        state = 2;
        p = nullptr;
    }

    // branch 3
    if (state == 1) {
        *p = 42; // Null dereference?
    }
}
```

⊗ 0 ⚠ 0 -- NORMAL -- UTF-8 CRLF C++

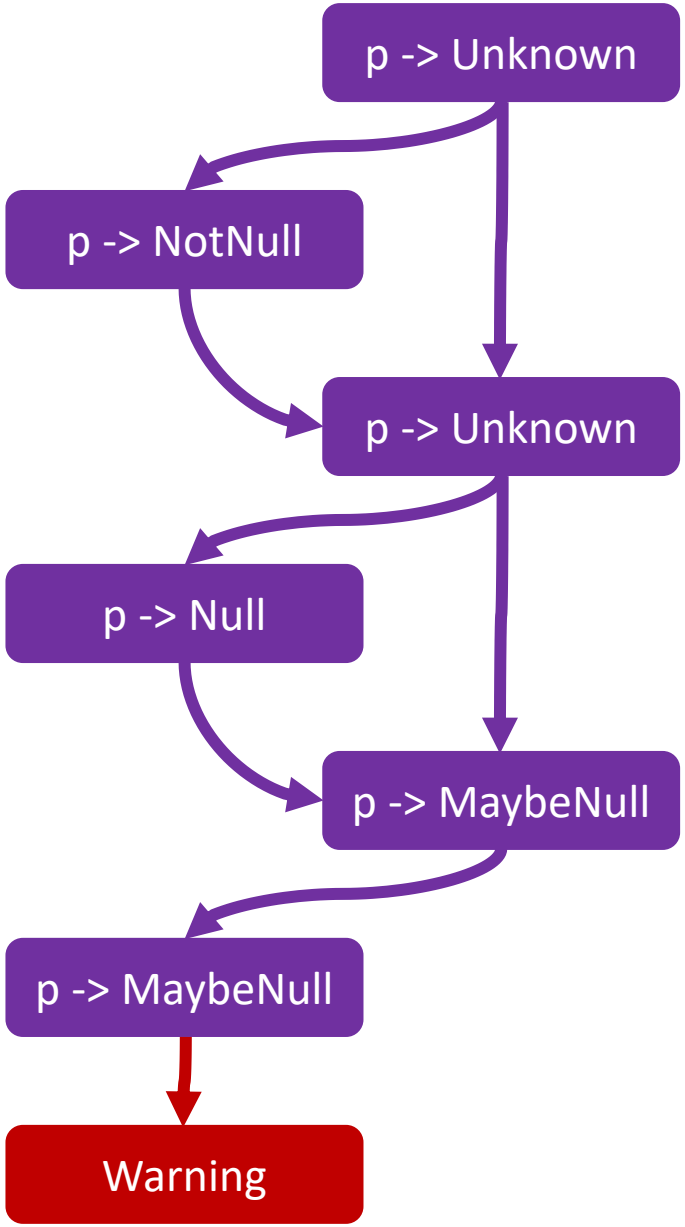
Flow-sensitive checks

```
void flow_sensitive(int *p, bool cond) {
  int var = 0;

  // branch 1
  if (p != nullptr) {
    var = 1;
  }

  // branch 2
  if (cond) {
    var = 2;
    p = nullptr;
  }

  // branch 3
  if (var == 1) {
    *p = 42; // Null dereference?
  }
}
```

Transition semi-lattice



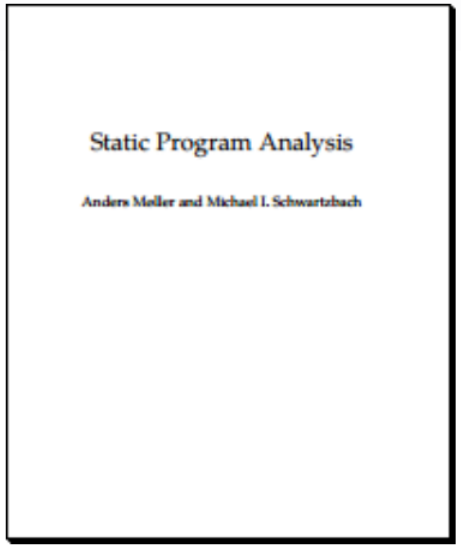
Analysis state



```
void path_sensitive(int *p, bool cond) {
  int var = 0;

  // branch 1
  if (p != nullptr) {
    var = 1;
  }

  // branch 2
  if (cond) {
    var = 2;
    p = nullptr;
  }

  // branch 3
  if (var == 1) {
    *p = 42; // Null dereference?
  }
}
```

- Some paths are infeasible:
  - Not taking branch 1, but taking branch 3
- All warnings on infeasible paths are noise
- Need info on the whole state!

Flow-sensitive analysis resources



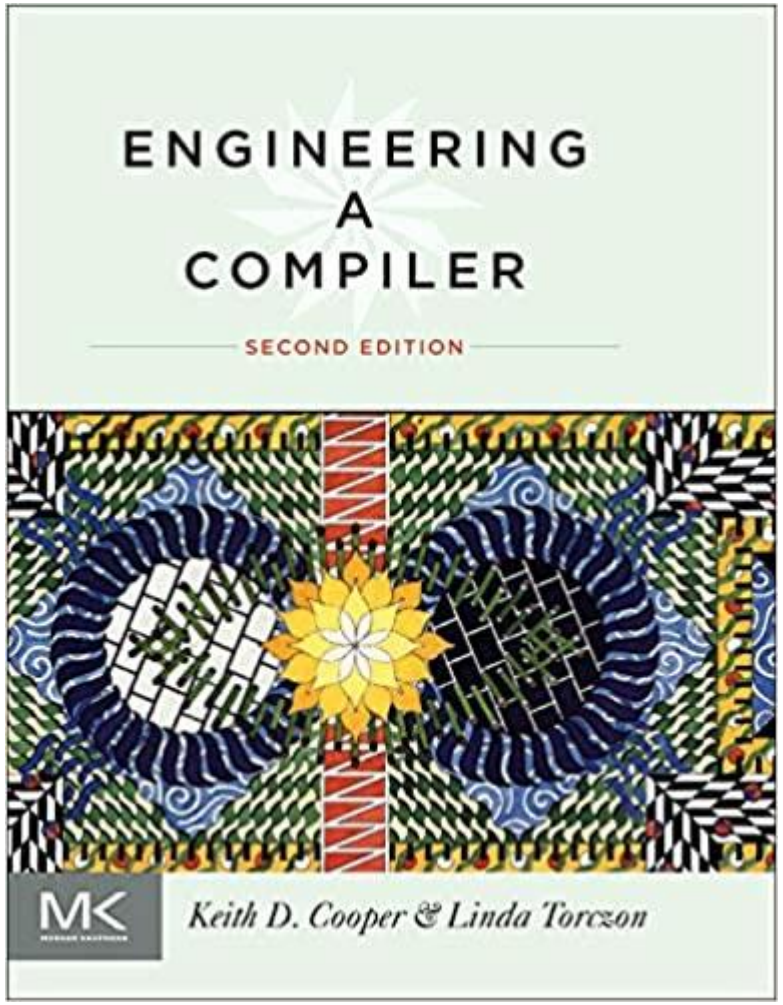
Static Program Analysis

Anders Møller and Michael I. Schwartzbach
Department of Computer Science, Aarhus University

Last revision: June 2021

PDF

BibTeX



File Edit Selection ··· pathsensitive.cpp - Visu...

pathsensitive.cpp

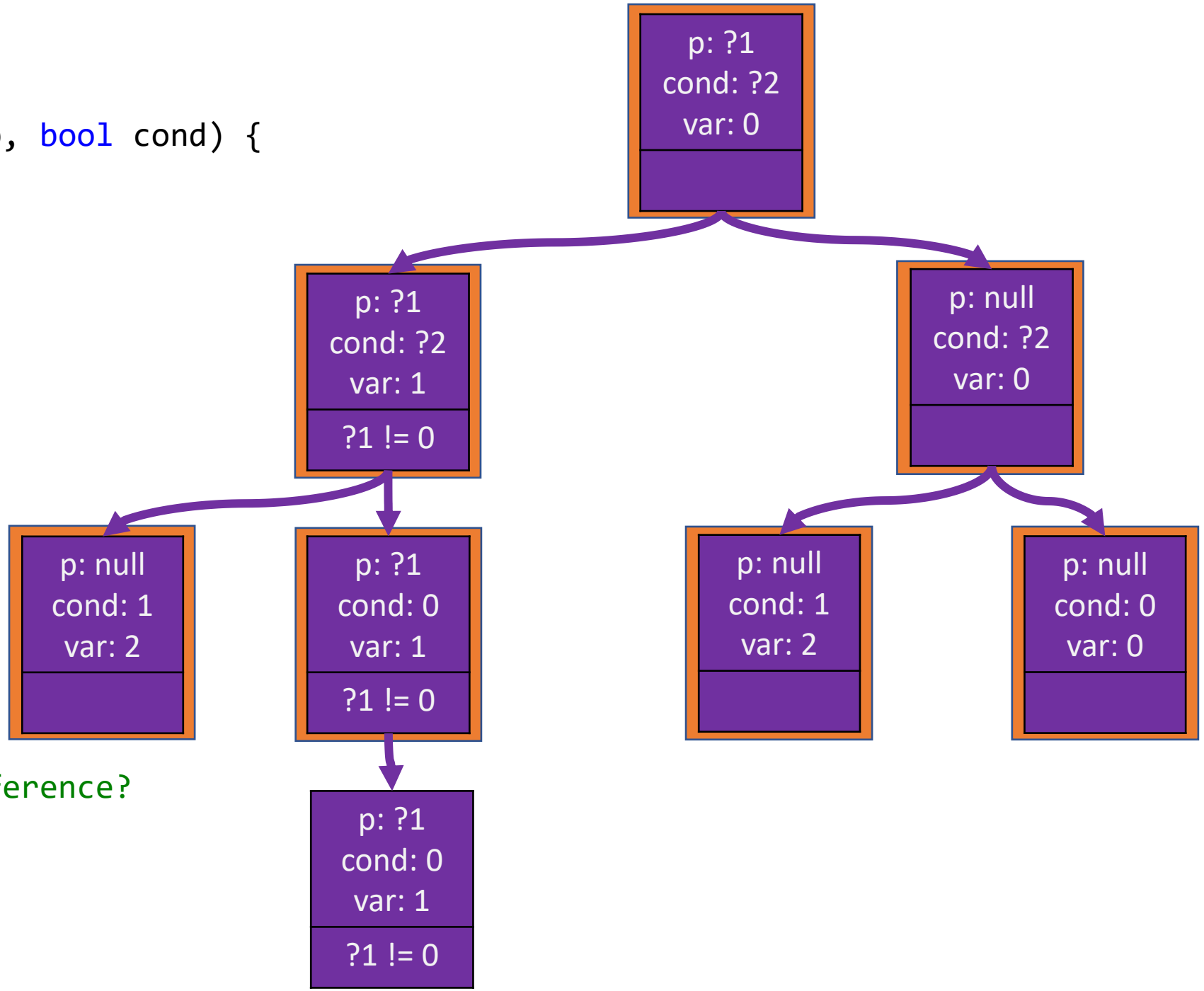
Drive - Microsoft > Documents > pathsensitive.cpp > path_sensitive(int *

```cpp
void path_sensitive(int *p, bool cond) {
    int state = 0;

    // branch 1
    if (p != nullptr) {
        state = 1;
    }

    // branch 2
    if (cond) {
        state = 2;
        p = nullptr;
    }

    // branch 3
    if (state == 1) {
        *p = 42; // Null dereference?
    }
}
```

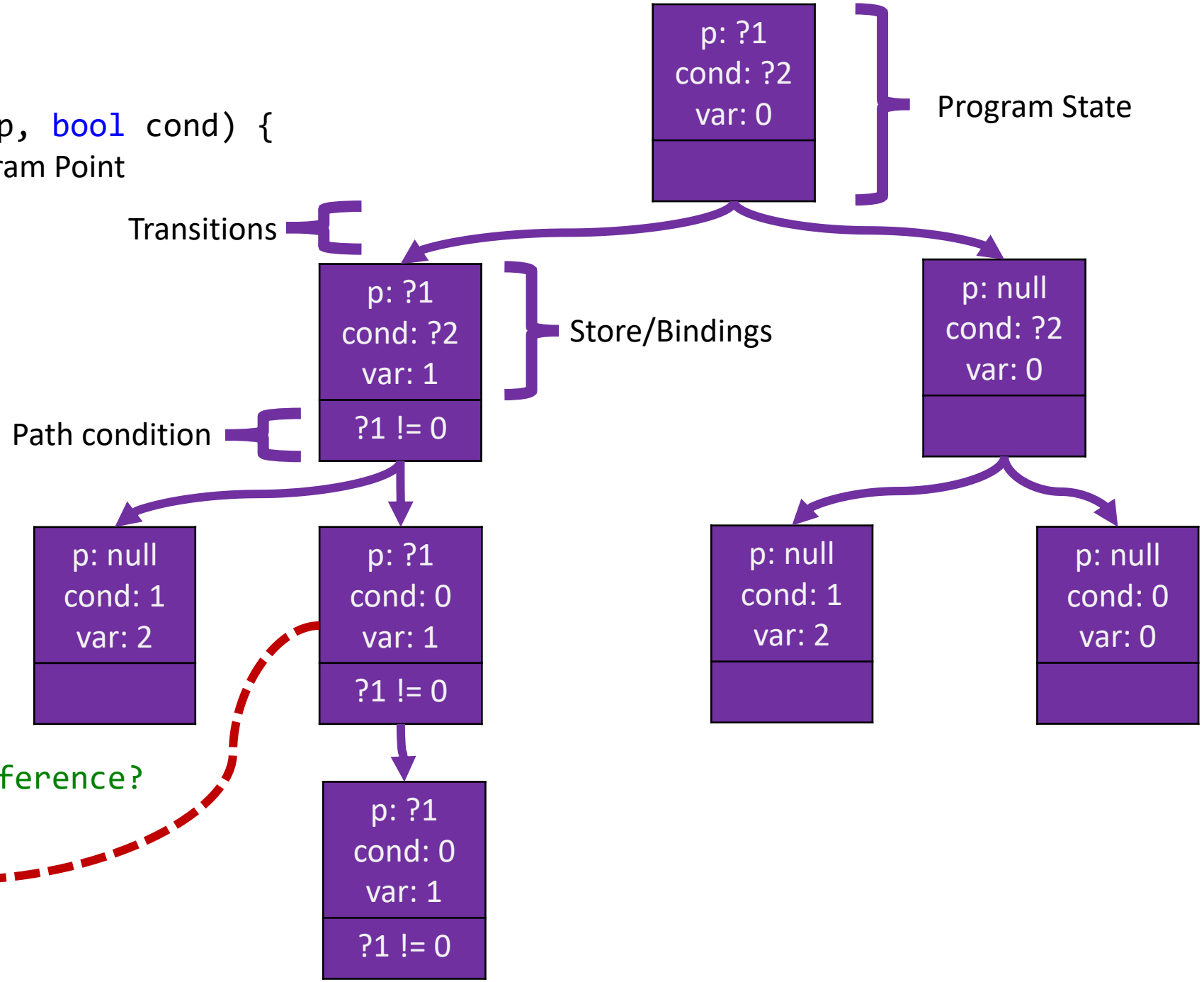
⊗ 0 ⚠ 0 -- NORMAL -- UTF-8 CRLF C++

Path-sensitive checks

```
void path_sensitive(int *p, bool cond) {
  int var = 0;

  // branch 1
  if (p != nullptr) {
    var = 1;
  } else {}

  // branch 2
  if (cond) {
    var = 2;
    p = nullptr;
  }

  // branch 3
  if (var == 1) {
    *p = 42; // Null dereference?
  }
}
```



```
void path_sensitive(int *p, bool cond) {
  int var = 0;

  // branch 1
  if (p != nullptr) {
    var = 1;
  }

  // branch 2
  if (cond) {
    var = 2;
    p = nullptr;
  }

  // branch 3
  if (var == 1) {
    *p = 42; // Null dereference?
  }
}
```



Performance

Precision

ESP: Path-Sensitive Program Verification in Polynomial Time

Path-Sensitive Dataflow Analysis with Iterative Refinement

MSVC has both

Path-sensitive

- Use after move
- Concurrency checks
- Variant clear
- Enum Index
- HResult
- Nullptr dereference
- more to come..

Flow-sensitive

- Coroutine lifetime checks
- Most of CppCoreChecks
  - Pointer safety analysis
  - Ownership analysis
  - Lifetime analysis (preview)
- more to come…

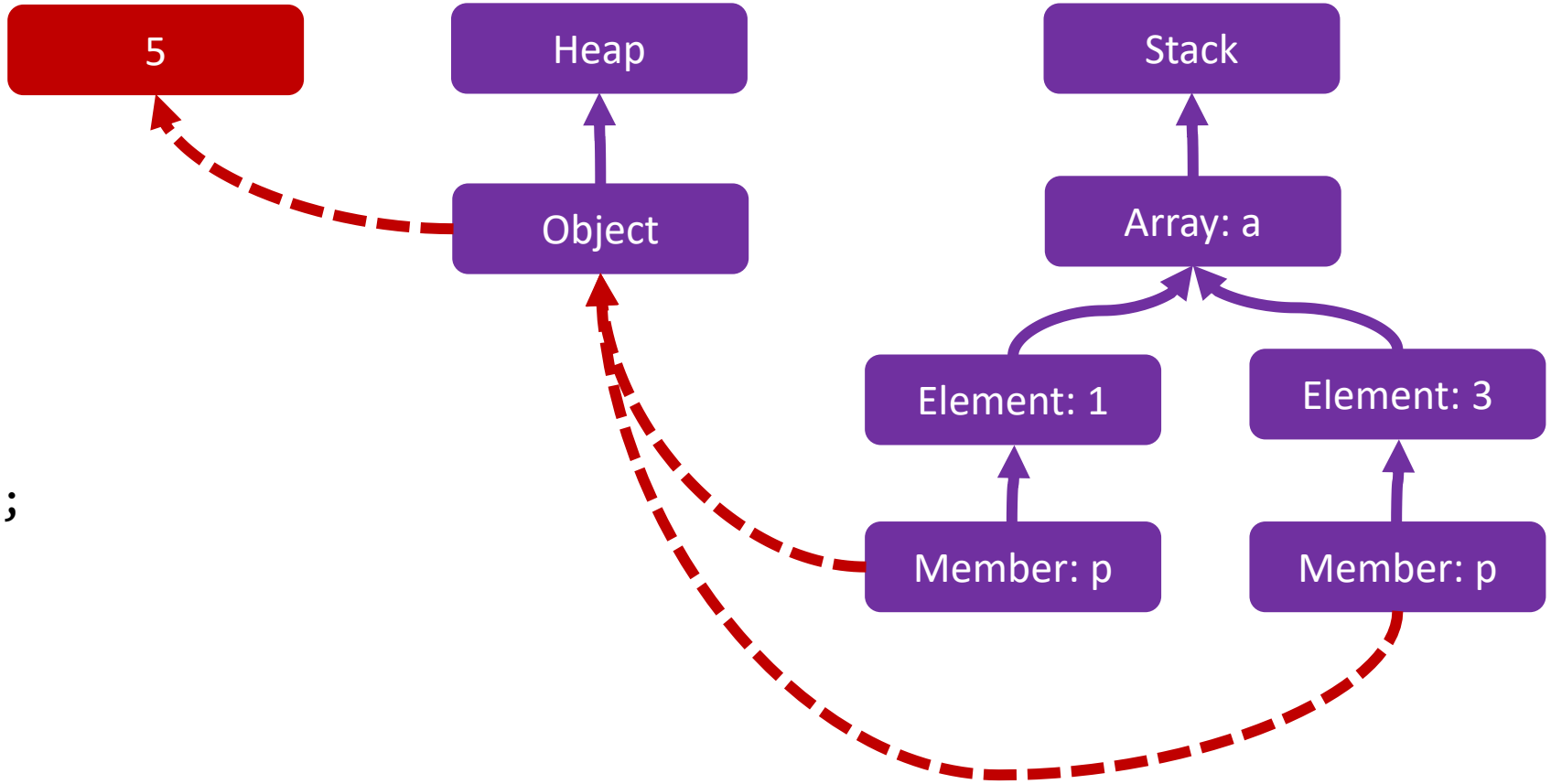
Looking under the hood

- Representing the store
- Solving constraints
- Guiding the analysis

- Immutable data structures?
- Modeling loops?

From source to values

```
struct S
{
    int *p;
};

int f()
{
    S a[10];
    // ...
    a[1].p = a[3].p;
    *a[3].p = 5;
}
```



MemoryRegion -> Values

From source to values

```
struct S
{
    int *p;
};

int f()
{
    S a[10];
    // ...
    a[1].p = a[3].p;
    *a[3].p = 5;
}
```

AliasSet(**a[1].p**, a[3].p)

AliasMaster -> Values

Location	Value
*a[1].p	5

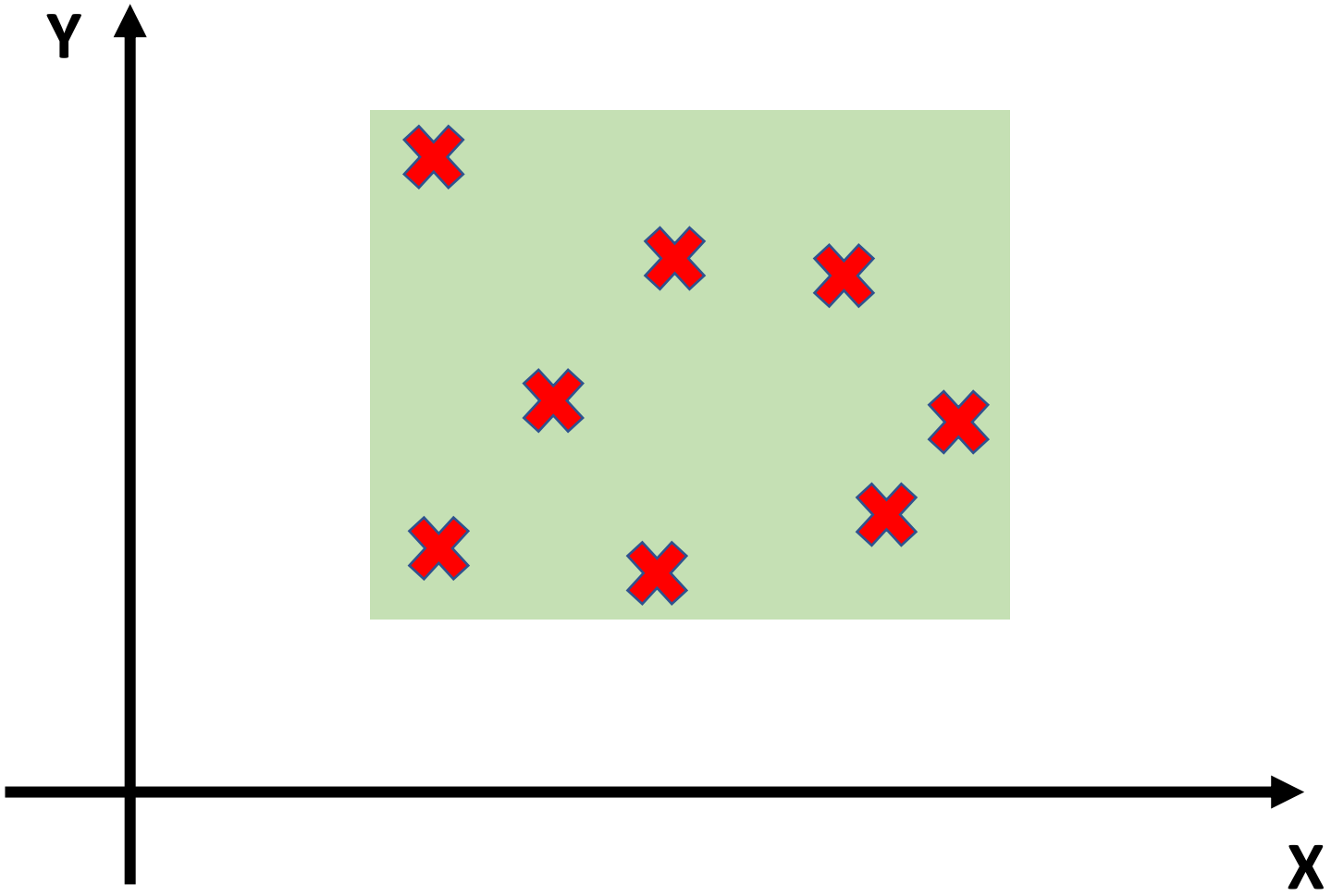
REPRESENTING ALIASES WITH ACCESS PATHS…

Fraunhofer IEM

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

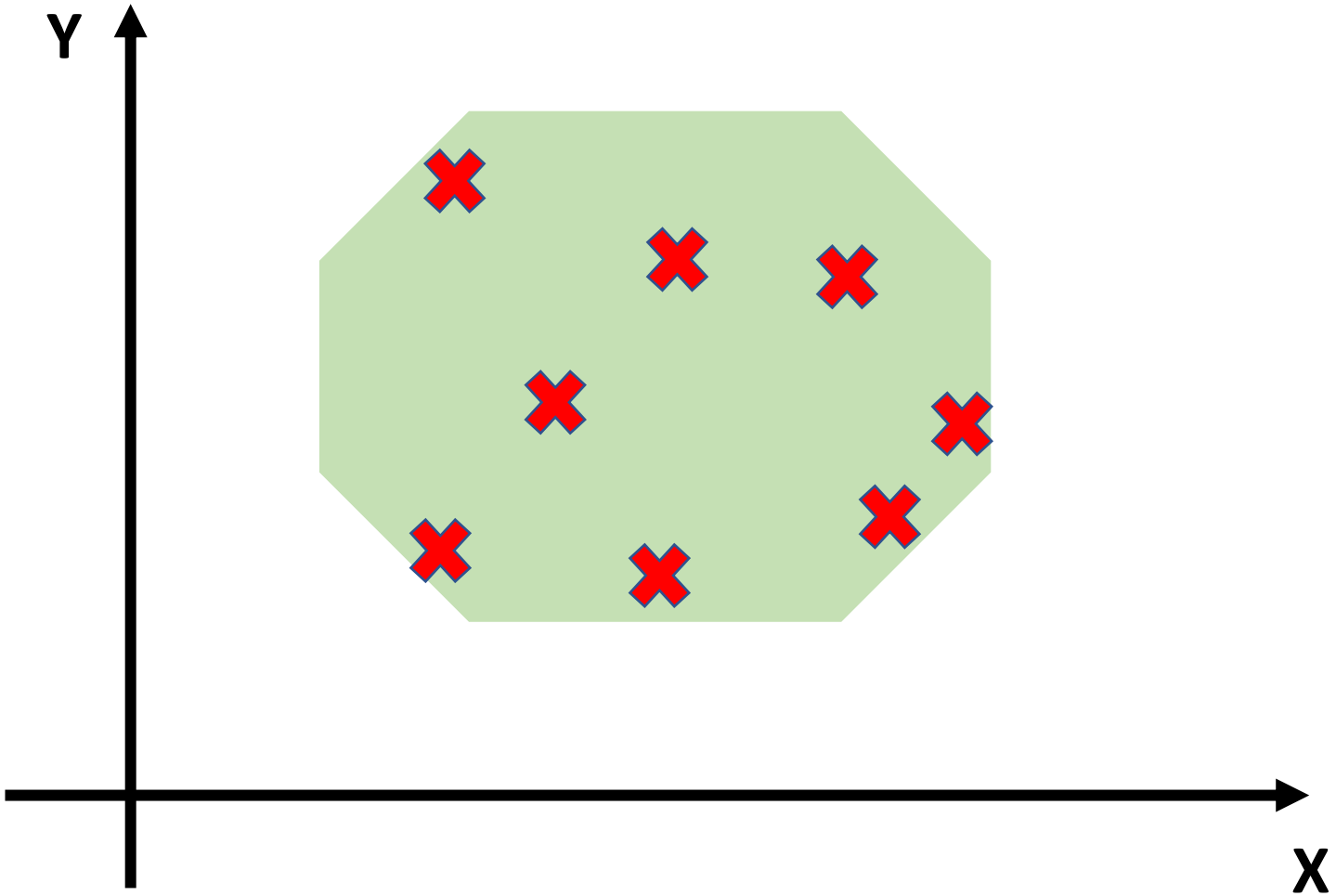
DECA I - Week 7 - c) Access paths

Intervals



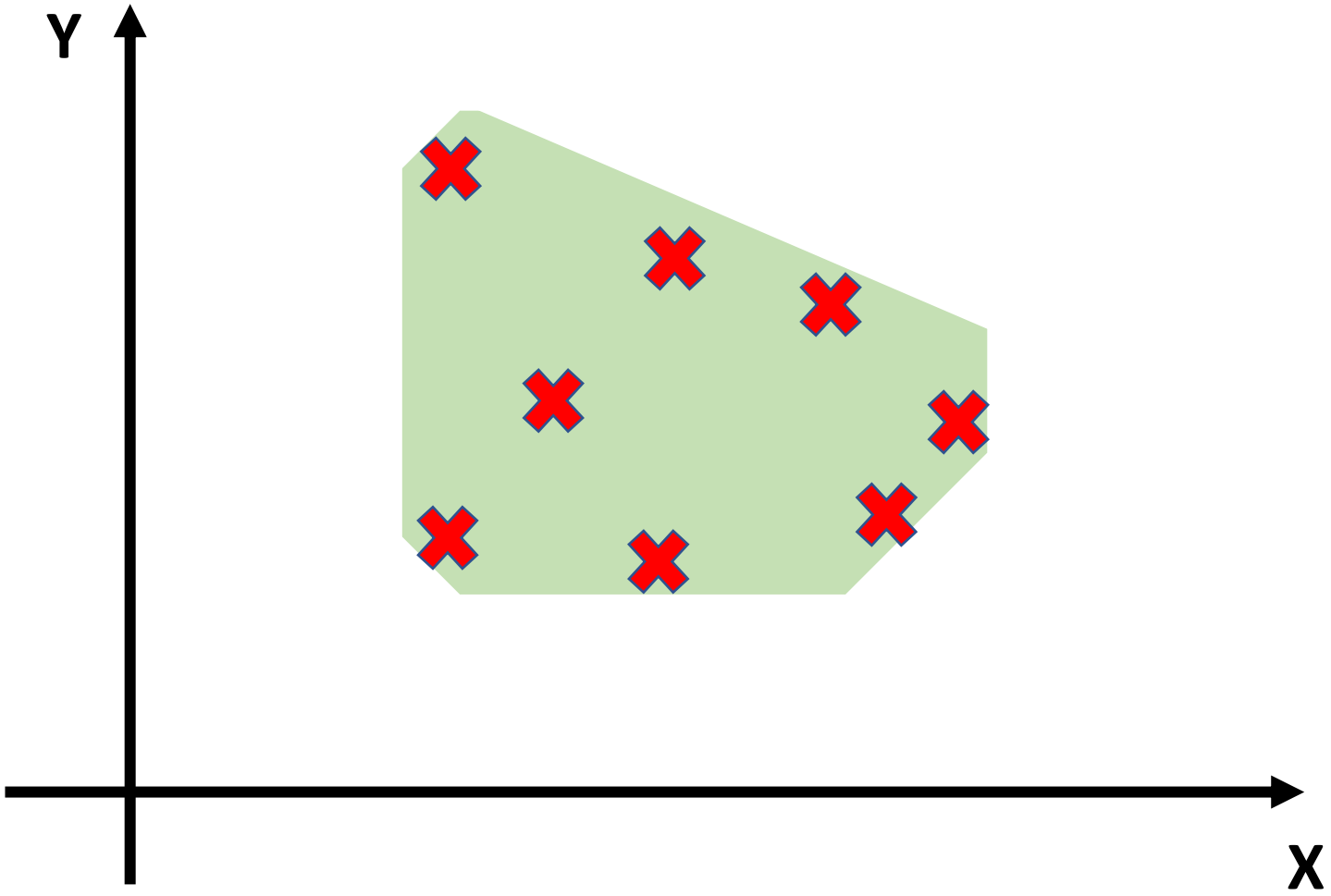
$$X \in [10, 20]$$
$$Y \in [10, 30]$$

Octagonal



$$10 \leq X \leq 20$$
$$10 \leq Y \leq 30$$
$$22 \leq X + Y \leq 47$$
$$25 \leq X - Y \leq 45$$

Polyhedra



$$10 \leq X \leq 20$$
$$10 \leq Y \leq 30$$
$$X + 2 * Y \leq 67$$
$$3{*}X - Y \leq 55$$

Queries

```
int a[10];

int f(int x, int y)
{
    if (x > y)
        return f(y, x);

    if (x < 0)
        return -1;

    if (y + x > 9)
        return 0;

    return a[y - x];
}
```

$x \leq y$

$x \geq 0$

$y + x \leq 9$

$0 \leq y - x \leq 9$

Given:

is

true?

$x \leq y$
$x \geq 0$
$y + x \leq 9$
$\boldsymbol{y - x < 0}$
Satisfiable?

$x \leq y$
$x \geq 0$
$y + x \leq 9$
$\boldsymbol{y - x > 9}$
Satisfiable?

Fourier-Motzkin elimination

Constraints

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq c$$

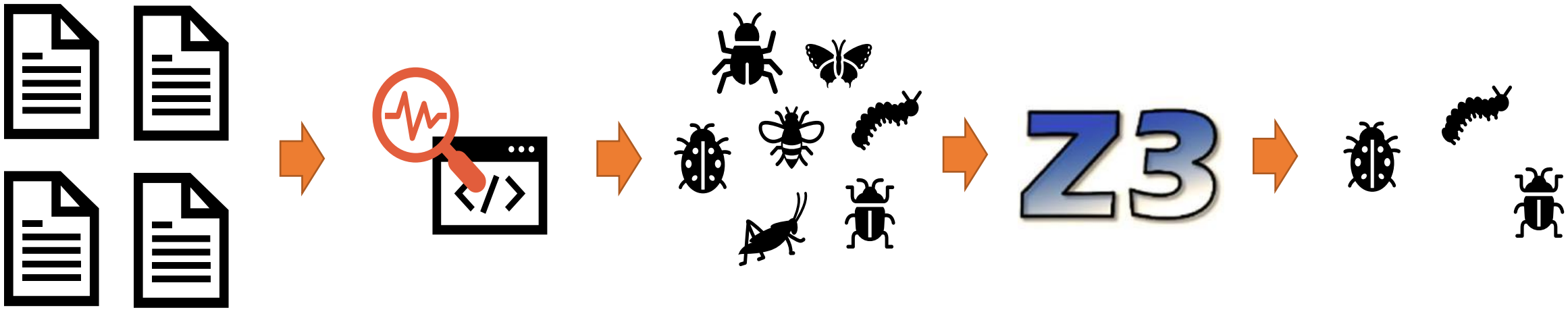
$$x \geq c \quad \Rightarrow \quad -x \leq -c$$

$$x = c \quad \Rightarrow \quad x \geq c,\ x \leq c \quad \Rightarrow \quad -x \leq -c, \quad x \leq c$$

$$x_1 \leq \mathrm{x}_2 \quad \Rightarrow \quad x_1 - x_2 \leq 0$$

$$x < \mathrm{c} \quad \Rightarrow \quad x + 1 \leq c$$

17 min -> 2h 54m 😱



```
1
2
3
4
5  bool isPositive(int x) {
6      return x > 0;
7  }
8
9  int foo(int *p, int x)
10 {
11     if (x < 0)
12         p = nullptr;
13
14     // ...
15     if (isPositive(x))
16         return *p;
17
18     return 42;
19 }
```

Analysis

```
C:\test>cl.exe /nologo /analyze:only /analyze:plugin EspXEngine.dll
 /analyze:ruleset "C:\test\talk\Nullptr.ruleset" talk\nullptr.cpp
nullptr.cpp

C:\test>
```

Lessons learned

- Path-sensitive analysis is powerful, but power comes at a cost
- We are actively looking into ways to make analysis more precise
- There are a large set of checks in MSVC
- Making intent explicit can help analysis, improve readability

Enjoy the rest of the conference!

Join #visual_studio channel on CppCon Discord
https://aka.ms/cppcon/discord

- Meet the Microsoft C++ team
- Ask any questions
- Discuss the latest announcements

Take our survey
https://aka.ms/cppcon

Our Sessions

Monday 25th

- Implementing C++ Modules: Lessons Learned, Lessons Abandoned – Cameron DaCamara & Gabriel Dos Reis

Tuesday 26th

- Documentation in The Era of Concepts and Ranges – Sy Brand & Christopher Di Bella (Google)
- Static Analysis and Program Safety in C++: Making it Real – Sunny Chatterjee
- In-memory and Persistent Representations of C++ – Gabriel Dos Reis (online 27th)
- Extending and Simplifying C++: Thoughts on pattern Matching using `is` and `as – Herb Sutter

Wednesday 27th

- What's New in Visual Studio: 64-bit IDE, C++20, WSL 2, and more – Sy Brand & Marian Luparu

Thursday 28th

- C++20's <chrono> Calendars and Time Zones in MSVC – Miya Natsuhara
- An Editor Can Do That? Debugging Assembly Language and GPU Kernels in Visual Studio Code – Julia Reid
- Why does std::format do that? – Charlie Barto
- Finding bugs using path-sensitive static analysis – Gabor Horvath (online 29th)

Thanks!

Fourier-Motzkin elimination

Resources

Improved Null Pointer Dereference Detection in Visual Studio 2022 version 17.0 Preview 4

Gabor

October 8th, 2021

The C++ static analysis team is committed to making your C++ coding experience as safe as possible. We are adding richer code safety checks and addressing high impact customer feedback bugs posted on the C++ Developer Community page. Thank you for engaging with us and giving us great feedback on the past releases and early previews leading to this point. Below is the detailed overview of a new experimental code analysis check that can detect null pointer dereference errors, along with a comparison to an existing check that has the same purpose.

Improved Null Pointer Dereference Detection in Visual Studio 2022 version 17.0 Preview 4 - C++ Team Blog (microsoft.com)

Gábor Horváth

Algorithms from a Compiler Developer's Toolbox

Gábor Horváth

C++ now

2021
MAY 2-7

1