LLOR: Automated Repair of OpenMP Programs

VMCAI 2025

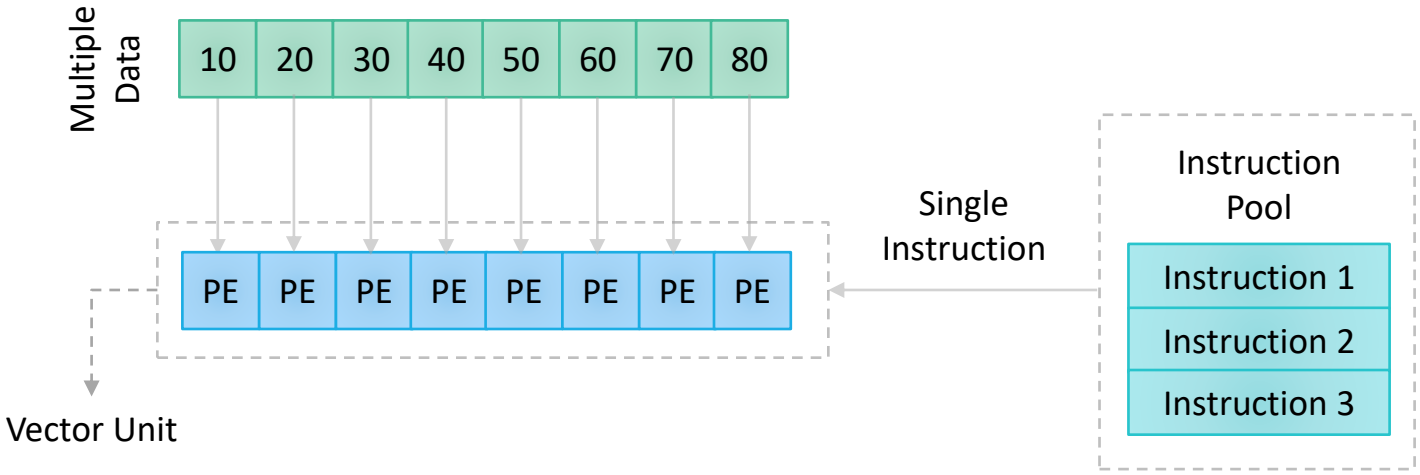
Authors

Utpal Bora (utpal.bora@cl.cam.ac.uk)
Saurabh Joshi (sbjoshi@iith.ac.in)
Gautam Muduganti (cs17resch01003@iith.ac.in)
Ramakrishna Upadrasta (ramakrishna@iith.ac.in)

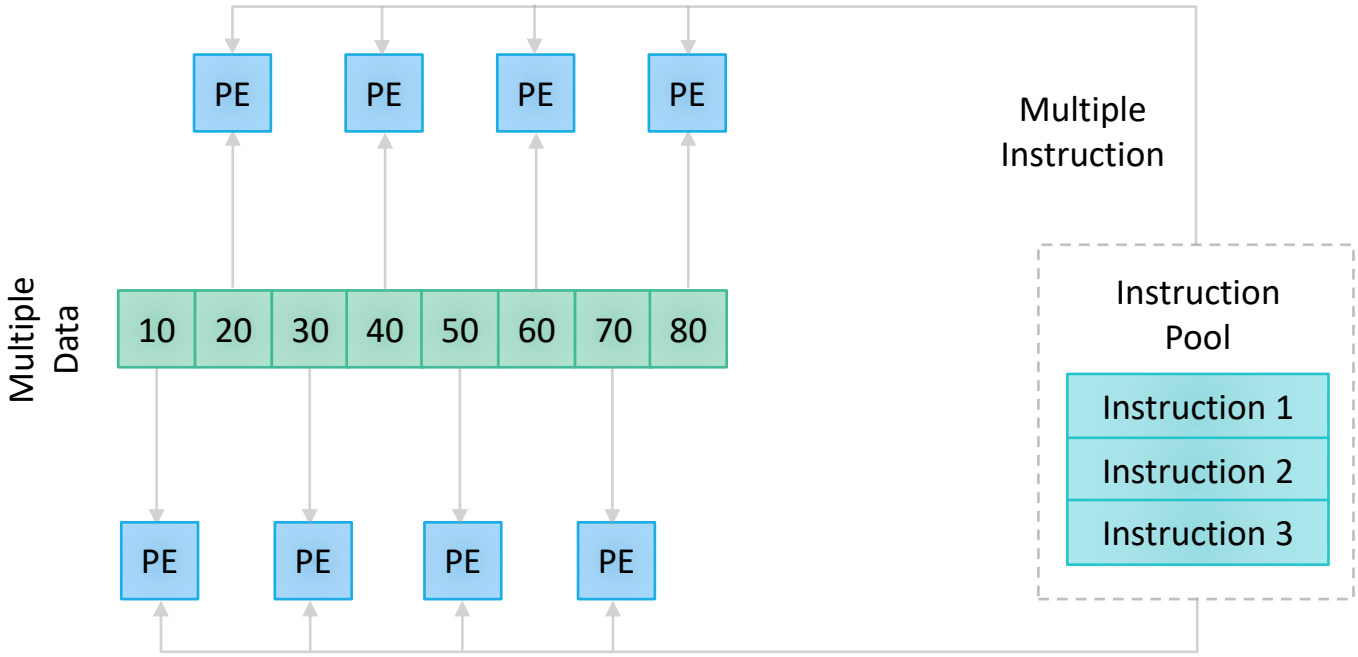
UNIVERSITY OF CAMBRIDGE

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Introduction to OpenMP (SIMD)



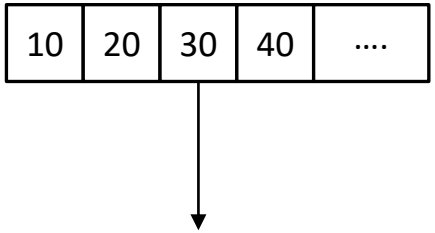
Introduction to OpenMP (MIMD)



Challenges of Parallel Programming: Data Races

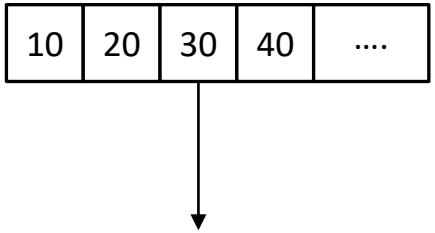
```
#pragma omp parallel {
    int temp = data[threadId + 1];
    data[threadId] = temp;
}
```

Challenges of Parallel Programming: Data Races



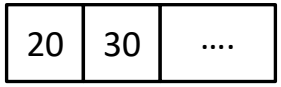

```
#pragma omp parallel {
    int temp = data[threadId + 1];
    data[threadId] = temp;
}
```

Challenges of Parallel Programming: Data Races

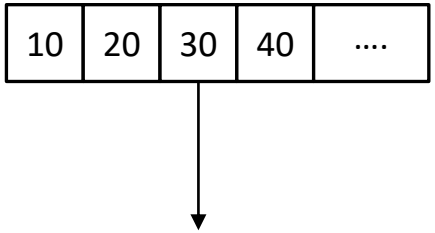



```
#pragma omp parallel {
    int temp = data[threadId + 1];
    data[threadId] = temp;
}
```

Thread-0	Thread-1
temp = 20;	temp = 30;
A[0] = 20;	A[1] = 30;

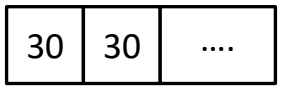


Challenges of Parallel Programming: Data Races

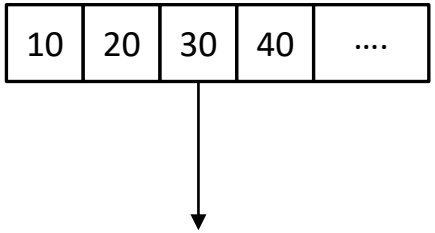



```
#pragma omp parallel {
    int temp = data[threadId + 1];
    data[threadId] = temp;
}
```

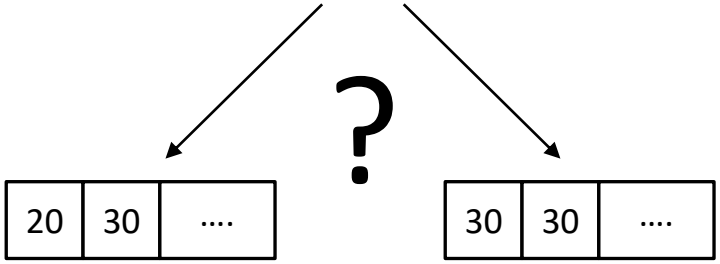
Thread-0	Thread-1
	temp = 30; A[1] = 30;
temp = 30; A[0] = 30;	



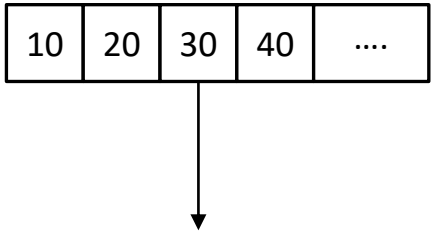
Challenges of Parallel Programming: Data Races



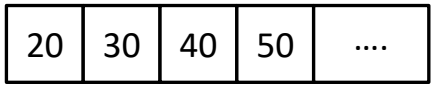

```
#pragma omp parallel {
    int temp = data[threadId + 1];
    data[threadId] = temp;
}
```



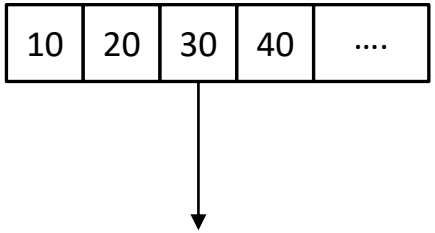
Challenges of Parallel Programming: Data Races



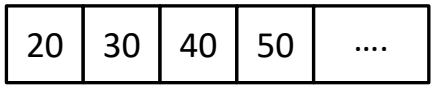

```
#pragma omp parallel {
    int temp = data[threadId + 1];
    #pragma omp barrier
    data[threadId] = temp;
}
```



Challenges of Parallel Programming: Data Races




```
#pragma omp parallel for ordered
for (int i=0; i<count; i++) {
    int temp = data[i + 1];
    #pragma omp ordered
    data[i] = temp;
}
```



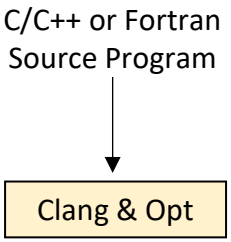
LLOR: Problem Statement

Given an OpenMP program P containing data race errors, LLOR returns a repaired program P' with the least number of synchronization constructs enabled in P'

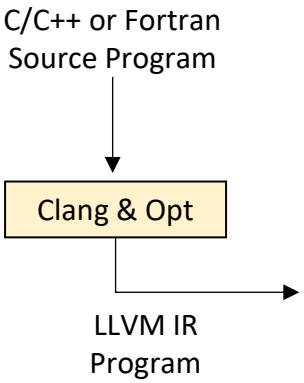
LLOR: Architecture

C/C++ or Fortran
Source Program

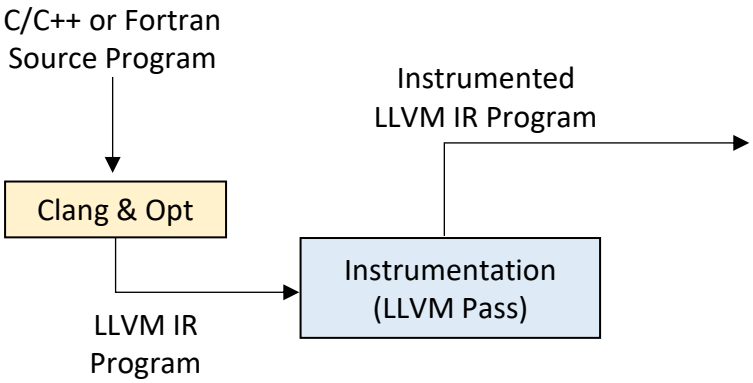
LLOR: Architecture



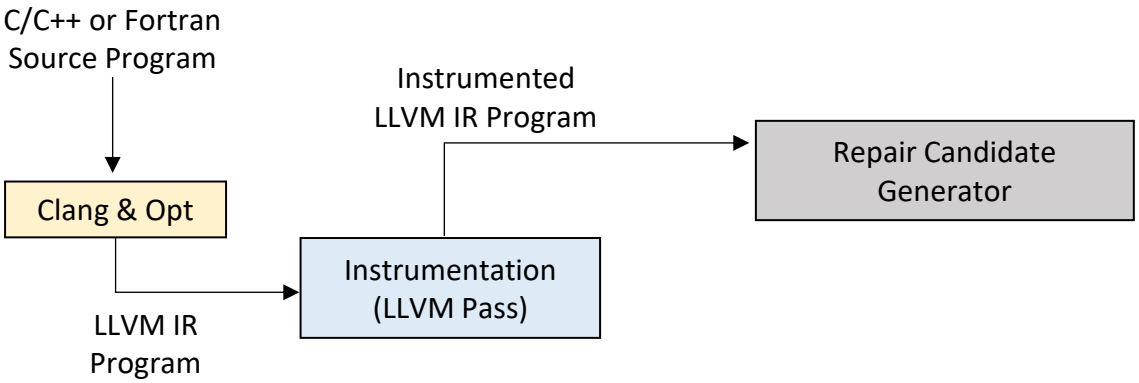
LLOR: Architecture



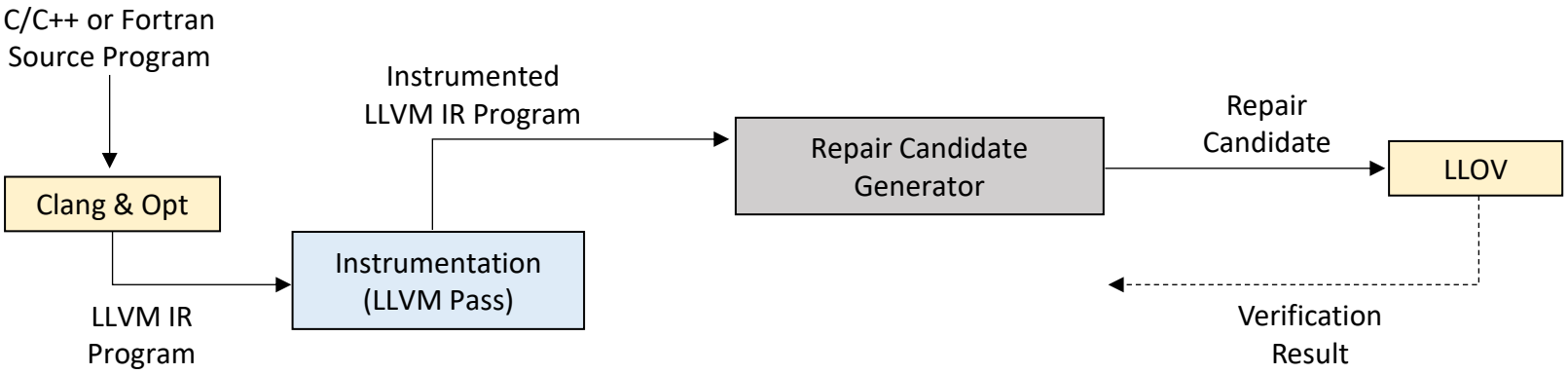
LLOR: Architecture



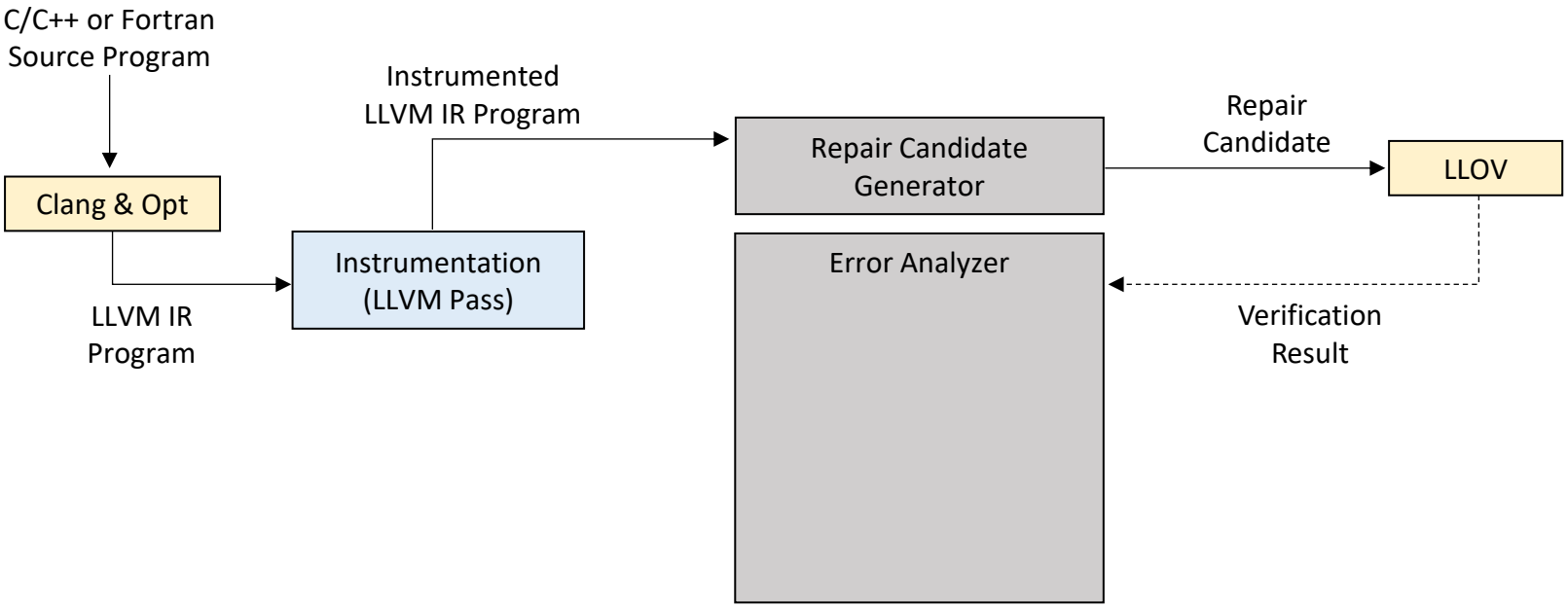
LLOR: Architecture



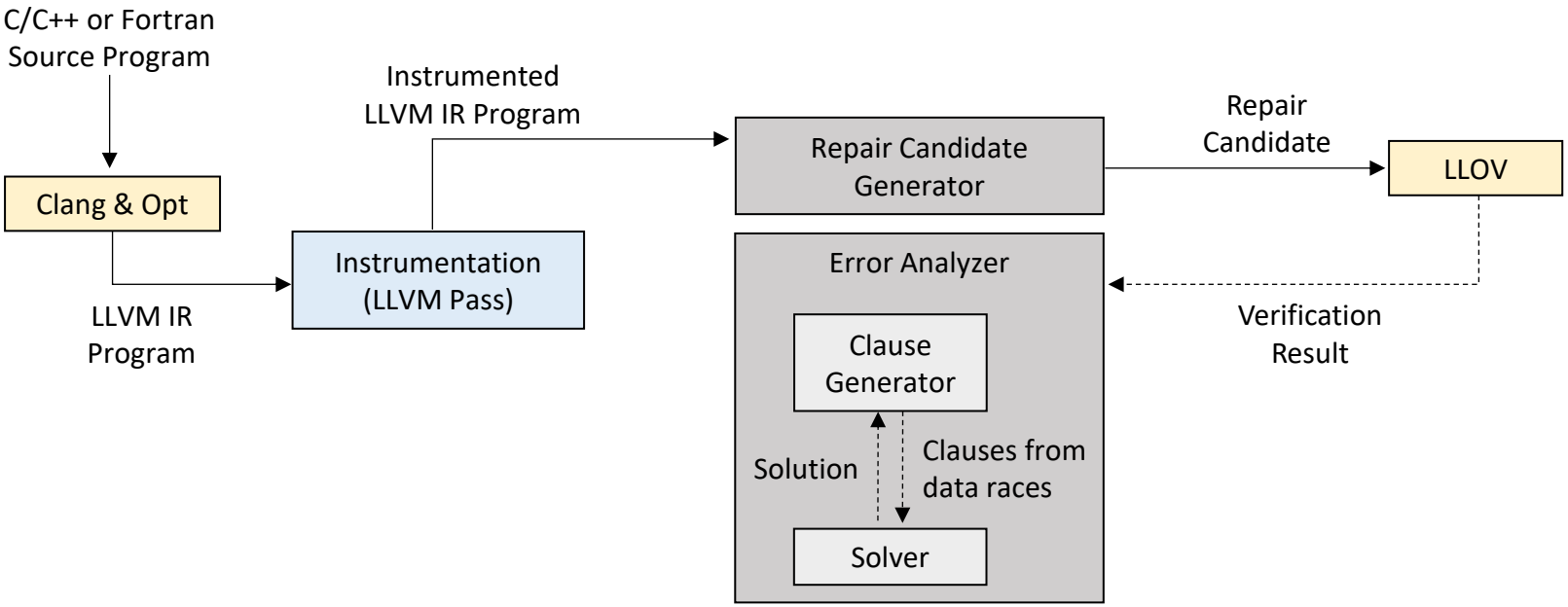
LLOR: Architecture



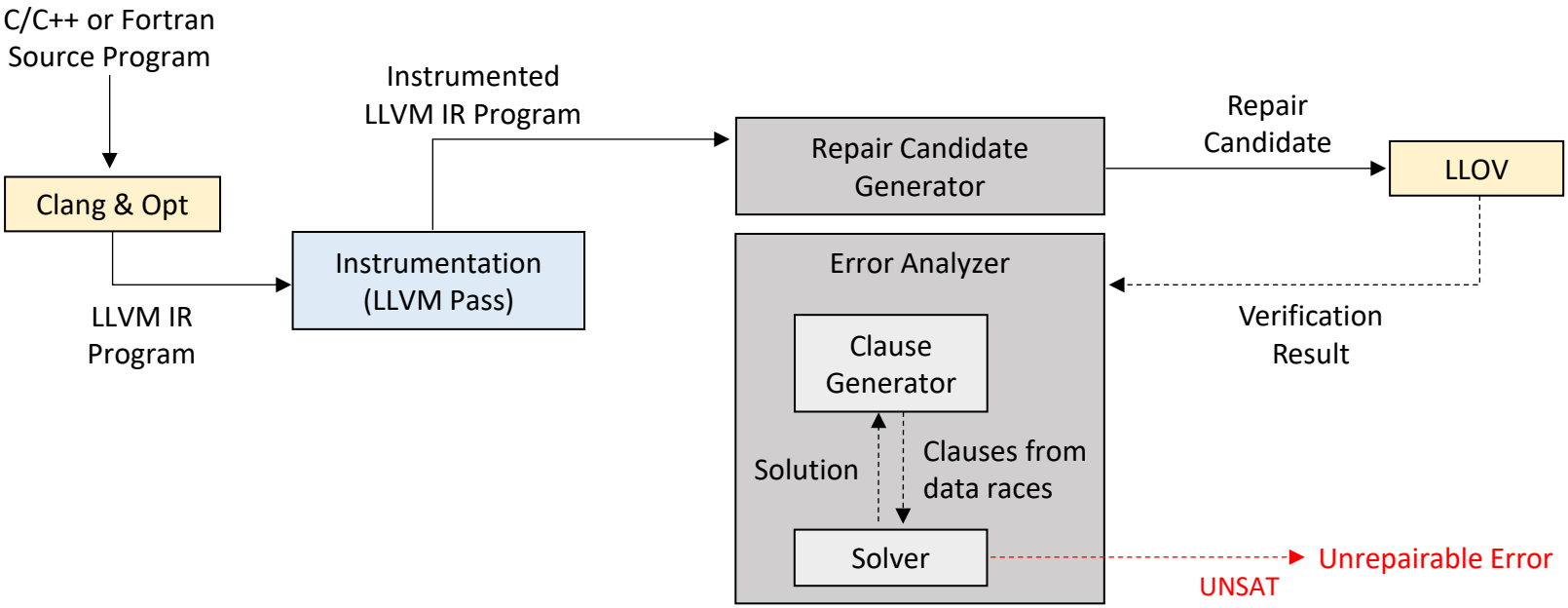
LLOR: Architecture



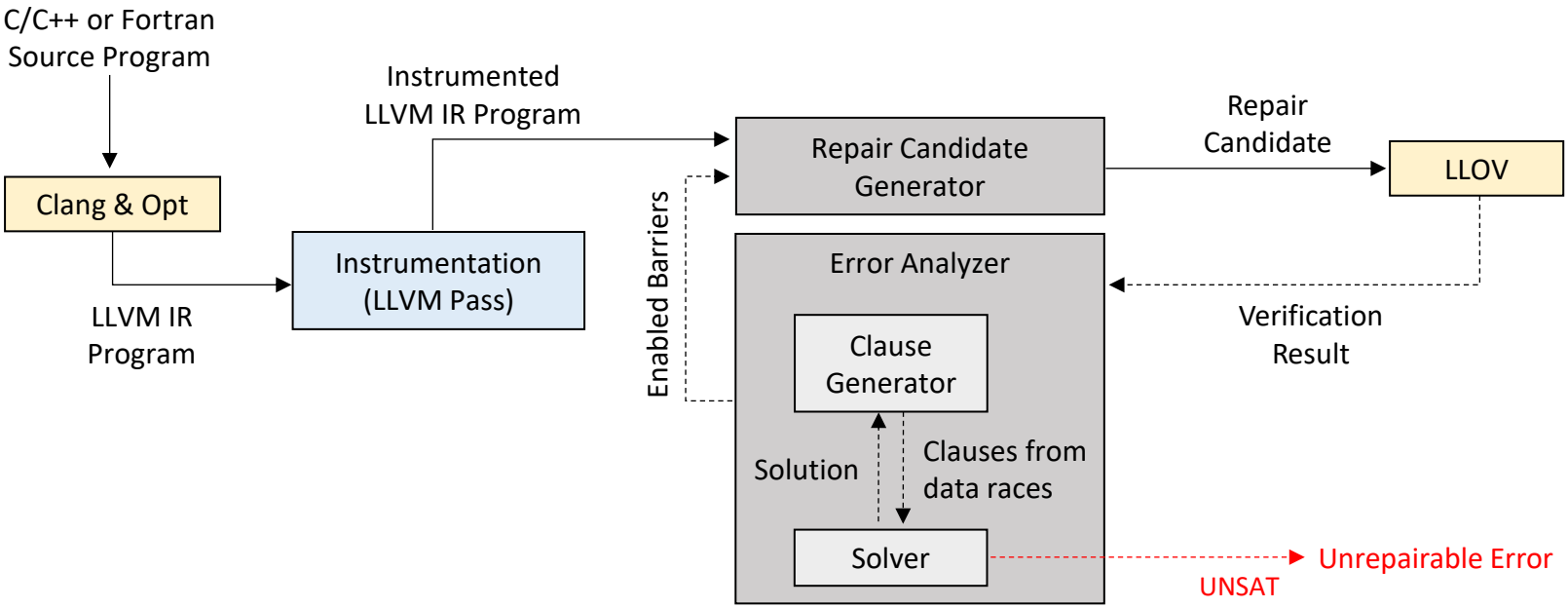
LLOR: Architecture



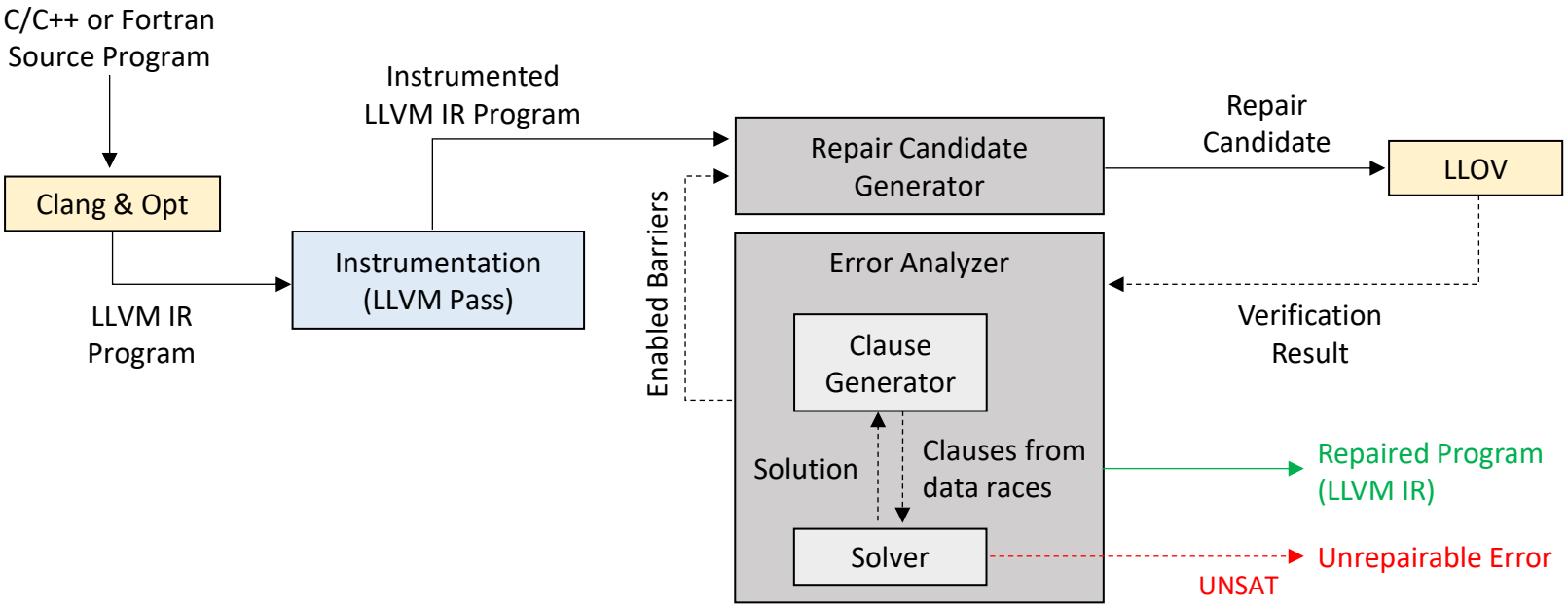
LLOR: Architecture



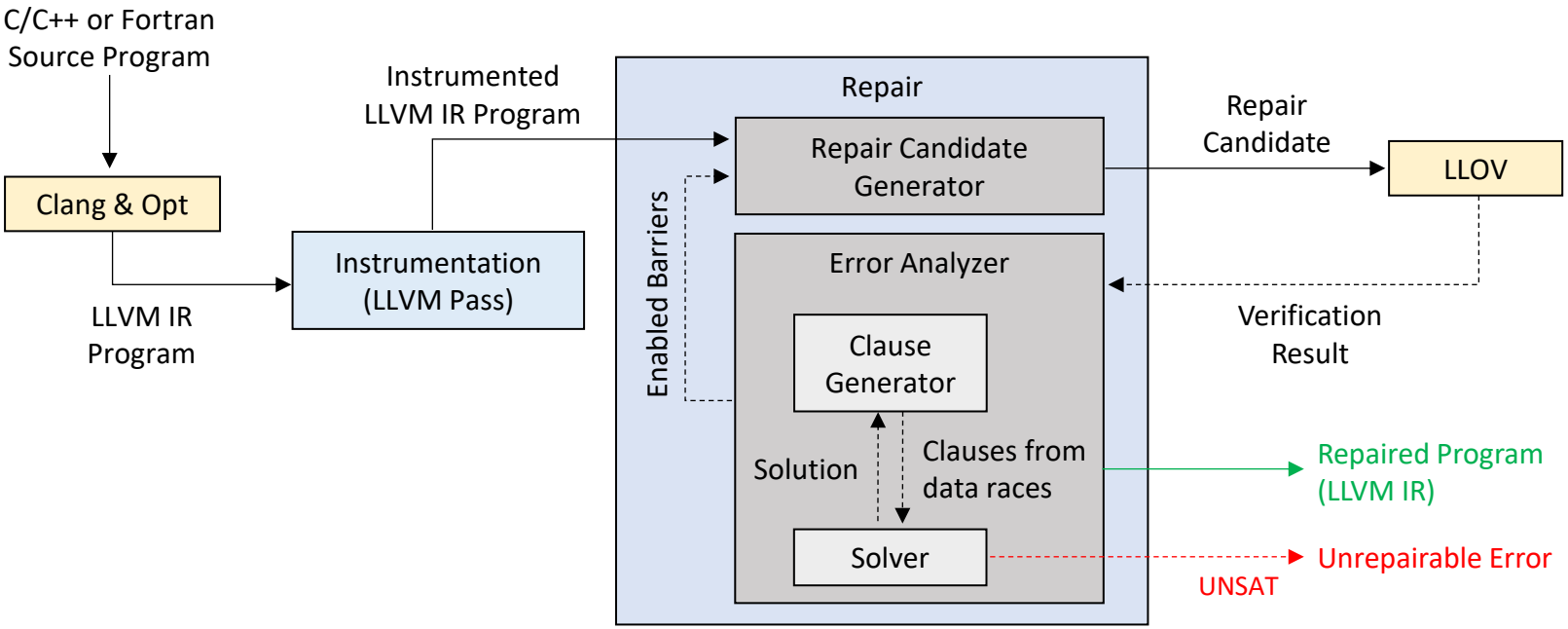
LLOR: Architecture



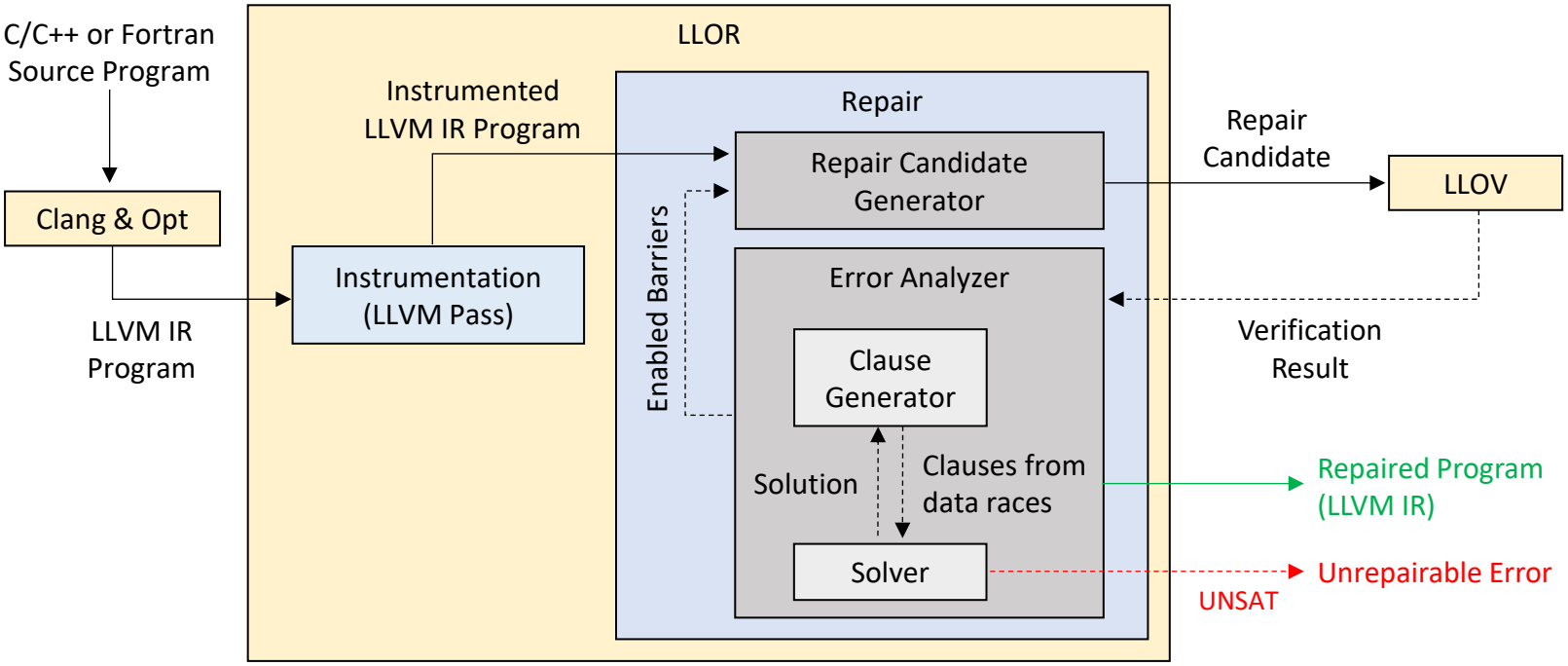
LLOR: Architecture



LLOR: Architecture

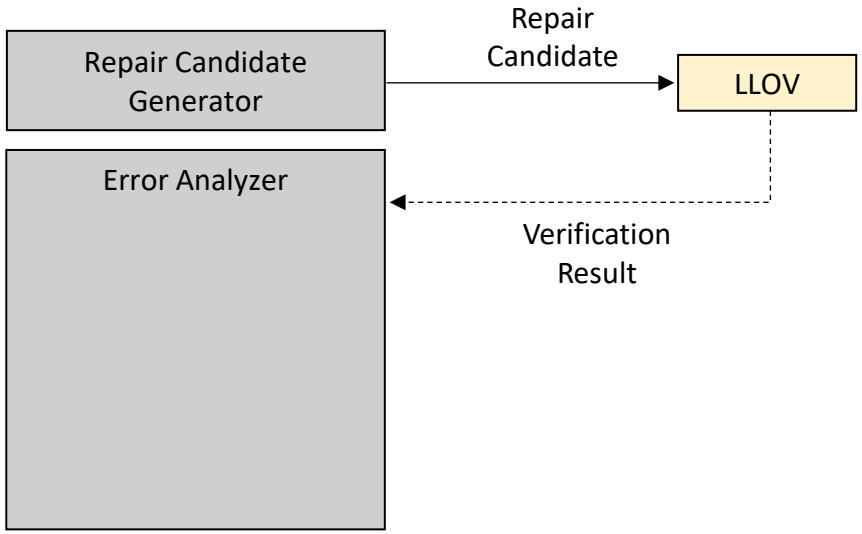


LLOR: Architecture



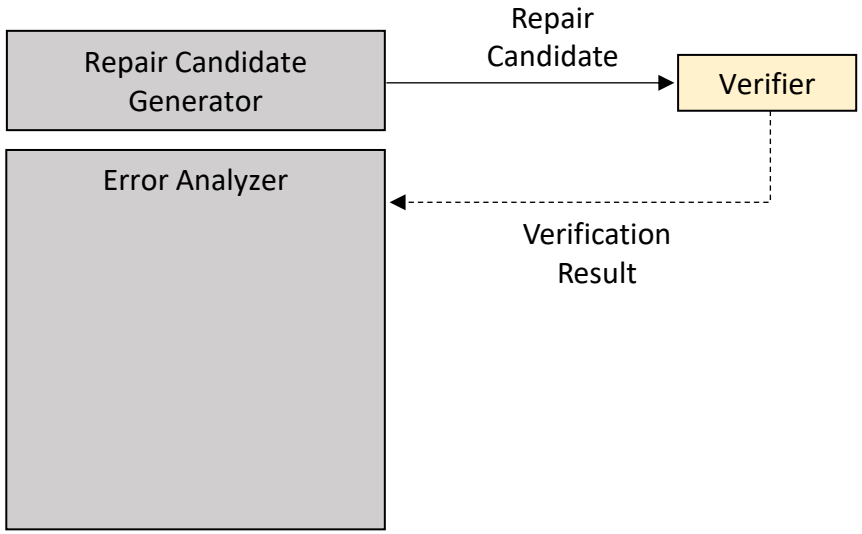
LLOR: Architecture

Counter-example guided repair synthesis approach



LLOR: Architecture

In principle, any verifier can be used as an oracle



Example (Parallel Regions)

```
#pragma omp parallel {
    int temp_a = data_a[threadId + 1];
    int temp_b = data_b[threadId + 1];

    data_a[threadId] = temp_a;
    data_b[threadId] = temp_b;
}
```

Example (Parallel Regions)

```
#pragma omp parallel {
    int temp_a = data_a[threadId + 1];
    int temp_b = data_b[threadId + 1];

    data_a[threadId] = temp_a;
    data_b[threadId] = temp_b;
}
```

Example (Parallel Regions)

```
#pragma omp parallel {
    int temp_a = data_a[threadId + 1];  // b1 = false;
    int temp_b = data_b[threadId + 1];  // b2 = false;

    data_a[threadId] = temp_a;          // b3 = false;
    data_b[threadId] = temp_b;          // b4 = false;
}
```

Example (Parallel Regions)

```
#pragma omp parallel {
    int temp_a = data_a[threadId + 1];  // b1 = false;
    int temp_b = data_b[threadId + 1];  // b2 = false;

    data_a[threadId] = temp_a;          // b3 = false;
    data_b[threadId] = temp_b;          // b4 = false;
}
```

Iteration 1

Clauses Generated:

$$b2 \vee b3$$

Solver Solution:

$$b2 \Leftrightarrow true$$

Example (Parallel Regions)

```
#pragma omp parallel {
    int temp_a = data_a[threadId + 1];   // b1 = false;
    #pragma omp barrier
    int temp_b = data_b[threadId + 1];   // b2 = true;

    data_a[threadId] = temp_a;           // b3 = false;
    data_b[threadId] = temp_b;           // b4 = false;
}
```

Example (Parallel Regions)

```
#pragma omp parallel {
    int temp_a = data_a[threadId + 1];   // b1 = false;
    #pragma omp barrier
    int temp_b = data_b[threadId + 1];   // b2 = true;

    data_a[threadId] = temp_a;           // b3 = false;
    data_b[threadId] = temp_b;           // b4 = false;
}
```

Iteration 2

Clauses Generated:

$$b2 \vee b3$$
$$b3 \vee b4$$

Solver Solution:

$$b3 \Leftrightarrow true$$

Example (Parallel Regions)

```
#pragma omp parallel {
    int temp_a = data_a[threadId + 1];  // b1 = false;
    int temp_b = data_b[threadId + 1];  // b2 = false;
    #pragma omp barrier
    data_a[threadId] = temp_a;          // b3 = true;
    data_b[threadId] = temp_b;          // b4 = false;
}
```

Example (Parallel For Loop)

```
#pragma omp parallel for
for (int i=0; i<count; i++) {
    int temp_a = data_a[i + 1];
    int temp_b = data_b[i + 1];

    data_a[i] = temp_a;
    data_b[i] = temp_b;
}
```

Example (Parallel For Loop)

```
#pragma omp parallel for
for (int i=0; i<count; i++) {
    int temp_a = data_a[i + 1];  // b1 = false;
    int temp_b = data_b[i + 1];  // b2 = false;

    data_a[i] = temp_a;          // b3 = false;
    data_b[i] = temp_b;          // b4 = false;
}
```

Example (Parallel For Loop)

```
#pragma omp parallel for
for (int i=0; i<count; i++) {
    int temp_a = data_a[i + 1];  // b1 = false;
    int temp_b = data_b[i + 1];  // b2 = false;

    data_a[i] = temp_a;          // b3 = false;
    data_b[i] = temp_b;          // b4 = false;
}
```

Clauses Generated:

$$b2 \vee b3$$
$$b3 \vee b4$$

Solver Solution:

$$b3 \Leftrightarrow true$$

Example (Parallel For Loop)

```
#pragma omp parallel for ordered
for (int i=0; i<count; i++) {
    int temp_a = data_a[i + 1];  // b1 = false;
    int temp_b = data_b[i + 1];  // b2 = false;
     #pragma omp ordered {
        data_a[i] = temp_a;      // b3 = true;
        data_b[i] = temp_b;      // b4 = false;
    }
}
```

Solver Strategies

1. MaxSAT Strategy

Solver Strategies

1. MaxSAT Strategy

Hard Clauses

$b1 \vee b2 \vee b4$
$b2 \vee b4$
$b2 \vee b3 \vee b5$
$b3 \vee b5 \vee b6$
$b1 \vee b6$

Soft Clauses

$\neg b1$
$\neg b2$
$\neg b3$
$\neg b4$
$\neg b5$
$\neg b6$

Solver Strategies

2. Minimal Hitting Set (mhs) Strategy
   - Polynomial-time greedy algorithm + Unit Literal Propagation

Solver Strategies

2. Minimal Hitting Set (mhs) Strategy
   - Polynomial-time greedy algorithm + Unit Literal Propagation

Example Clauses

$$b1 \vee b2 \vee b4$$
$$b2 \vee b4$$
$$b2 \vee b3 \vee b5$$
$$b3 \vee b5 \vee b6$$
$$b1 \vee b6$$

Solver Strategies

2. Minimal Hitting Set (mhs) Strategy
   - Polynomial-time greedy algorithm + Unit Literal Propagation

Example Clauses

$$b1 \vee b2 \vee b4$$
$$b2 \vee b4$$
$$b2 \vee b3 \vee b5$$
$$b3 \vee b5 \vee b6$$
$$b1 \vee b6$$

Solution

Solver Strategies

2. Minimal Hitting Set (mhs) Strategy
   - Polynomial-time greedy algorithm + Unit Literal Propagation

Example Clauses

$b1 \vee b2 \vee b4$
$b2 \vee b4$
$b2 \vee b3 \vee b5$
$b3 \vee b5 \vee b6$
$b1 \vee b6$

Solution

$b2 \Leftrightarrow true$

Solver Strategies

2. Minimal Hitting Set (mhs) Strategy
   - Polynomial-time greedy algorithm + Unit Literal Propagation

Example Clauses

$b1 \vee b2 \vee b4$
$b2 \vee b4$
$b2 \vee b3 \vee b5$
$b3 \vee b5 \vee b6$
$b1 \vee b6$

Solution

$b2 \Leftrightarrow true$
$b6 \Leftrightarrow true$

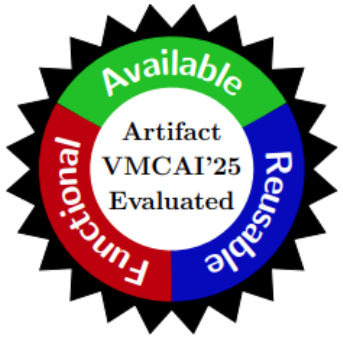
Benchmark Summary

Source	Programs		
	C/C++	Fortran	Total
DataRaceBench	181	168	349
Exascale Project	8	0	8
Rodinia	18	0	18
Parallel Research Kernels	11	0	11
Other Large Benchmarks	5	0	5
LLOR Test Suite	12	12	24
Total	**235**	**180**	**415**

Results

Category	C/C++	Fortran
Total benchmarks	235	180
I. No data races identified by LLOV	75	129
No changes made by LLOR	71	122
Changes recommended by LLOR	4	7
II. Data races identified by LLOV	117	30
Repaired by LLOR	92	15
Could not be repaired by LLOR	5	11
Timeouts (300 seconds)	9	0
Unsupported	11	4
III. Unsupported by LLOV	43	21
Unsupported by LLOR	37	11
Compilation errors	0	10
Verification errors	6	0

Try LLOR

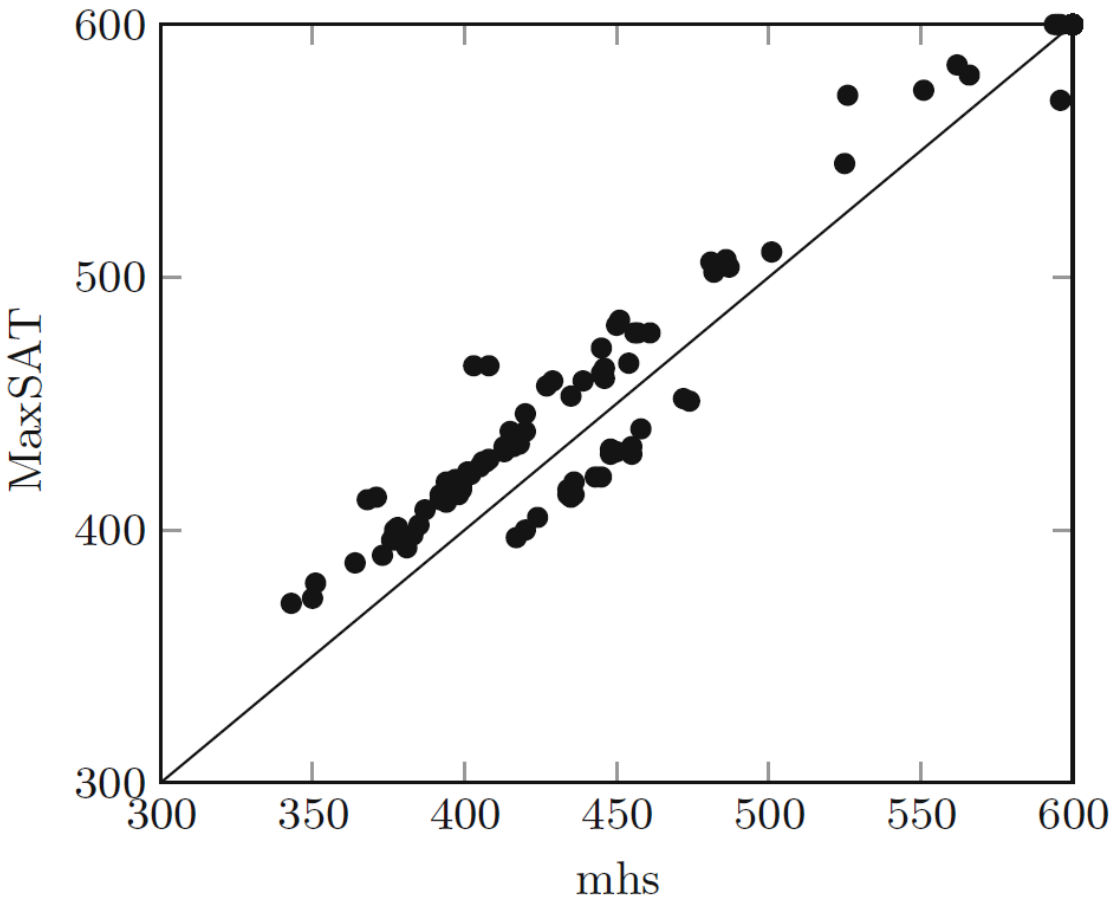


Paper: https://arxiv.org/abs/2411.14590

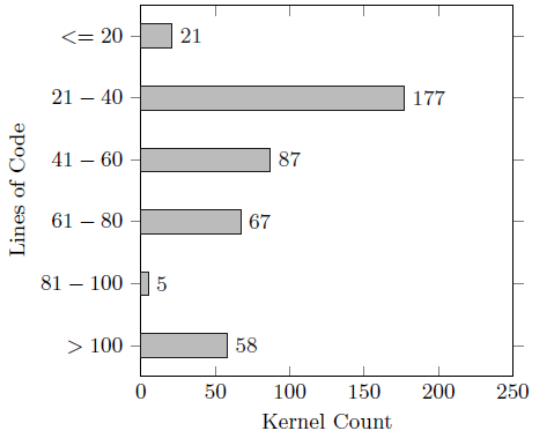
Source Code: https://github.com/cs17resch01003/llor

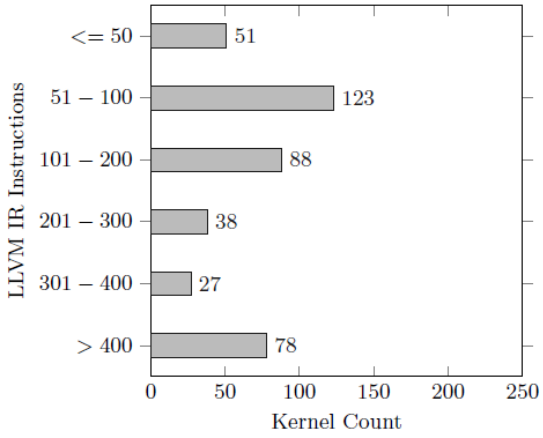
VMCAI 2025 Artifacts: https://zenodo.org/records/13938526

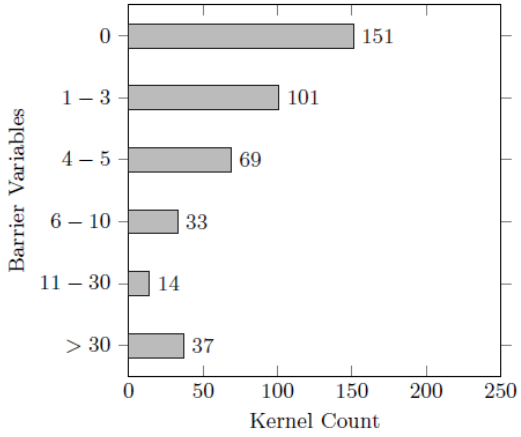
MaxSAT vs. mhs



MaxSAT vs. mhs







Pointer Data Race

```
static int* counter;
int main() {
    #pragma omp parallel
        (*counter)++
}
```

Pointer Data Race

```
static int* counter;
int main() {
    #pragma omp parallel
        (*counter)++
}
```

```
%3 = load i32*, i32** @counter
%4 = load i32, i32* %3
%5 = add nsw i32 %4, 1
store i32 %5, i32* %3
```

Pointer Data Race

```
static int* counter;
int main() {
    #pragma omp parallel
        (*counter)++
}
```

```
%3 = load i32*, i32** @counter
%4 = load i32, i32* %3
%5 = add nsw i32 %4, 1
store i32 %5, i32* %3
```

Pointer Data Race

```
static int* counter;
int main() {
    #pragma omp parallel
        (*counter)++
}
```

```
%3 = load i32*, i32** @counter
%4 = load i32, i32* %3
%5 = add nsw i32 %4, 1
store i32 %5, i32* %3
```

Write-Write Conflict

```
int data[NUM_THREADS+1];
#pragma omp parallel {
    int id = omp_get_thread_num();
    data[0] = id;
}
```