FastKLEE: Faster Symbolic Execution via Reducing Redundant Bound Checking of Type-Safe Pointers

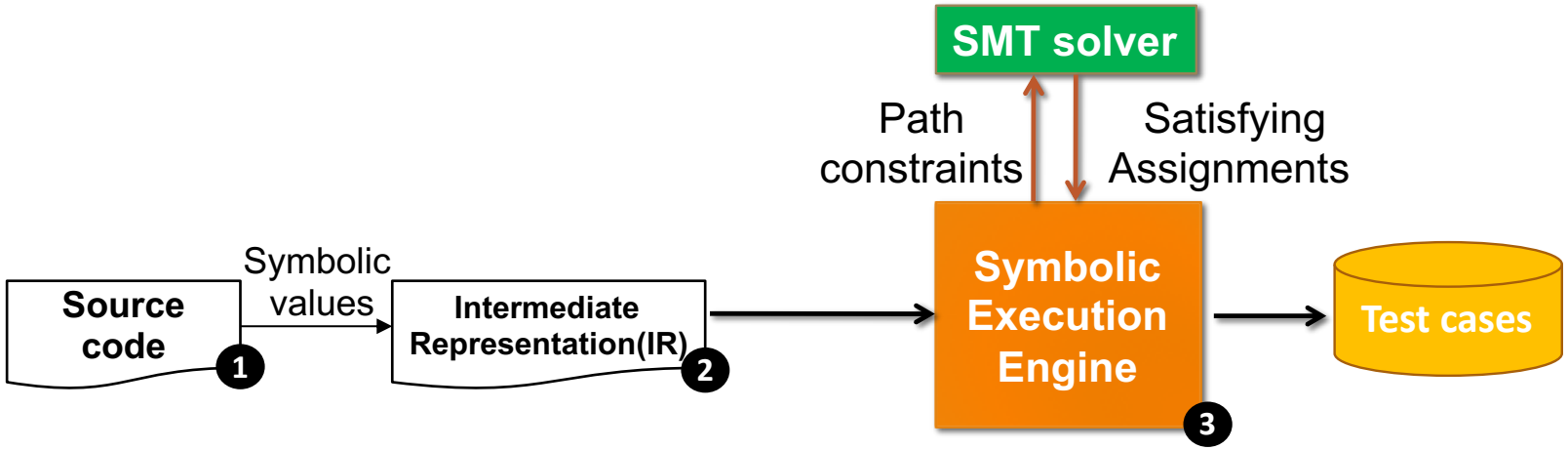
Haoxin Tu, Lingxiao Jiang, Xuhua Ding (Singapore Management University)
He Jiang (Dalian University of Technology)

(Accepted work on Tool Demonstration of FSE 2022)

15/04/2024, Lisbon

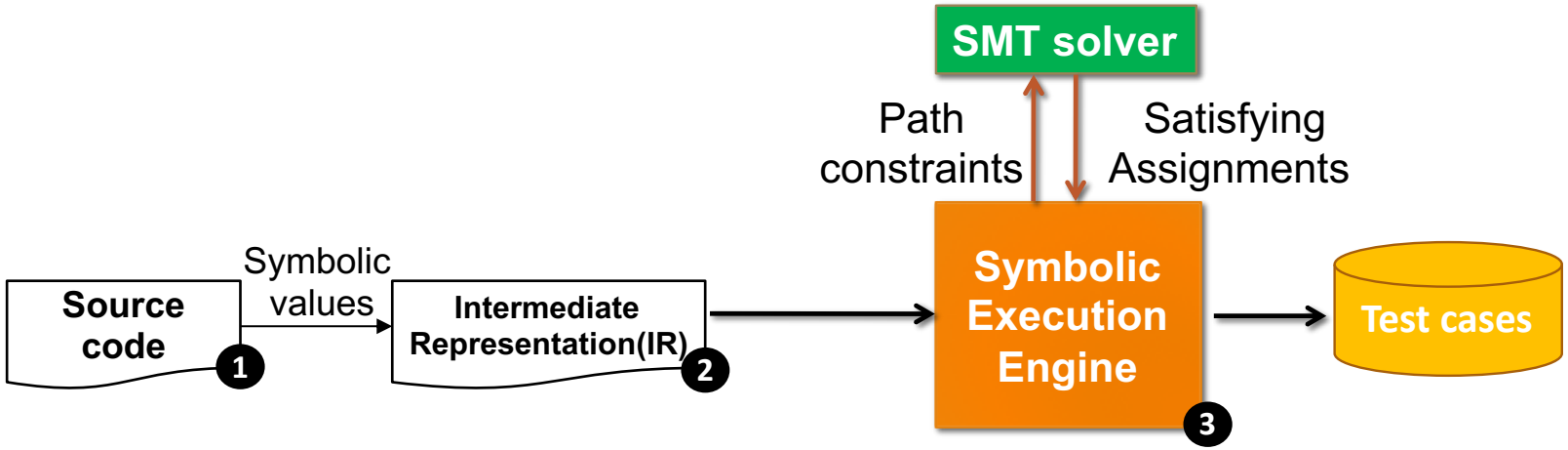
Background

- **Symbolic execution is popular**
- **General workflow of traditional symbolic execution engine (e.g., KLEE)**



Background

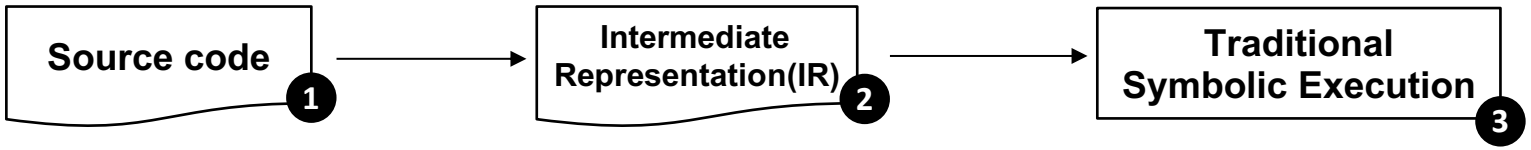
- **Symbolic execution is popular**
- **General workflow of traditional symbolic execution engine (e.g., KLEE)**



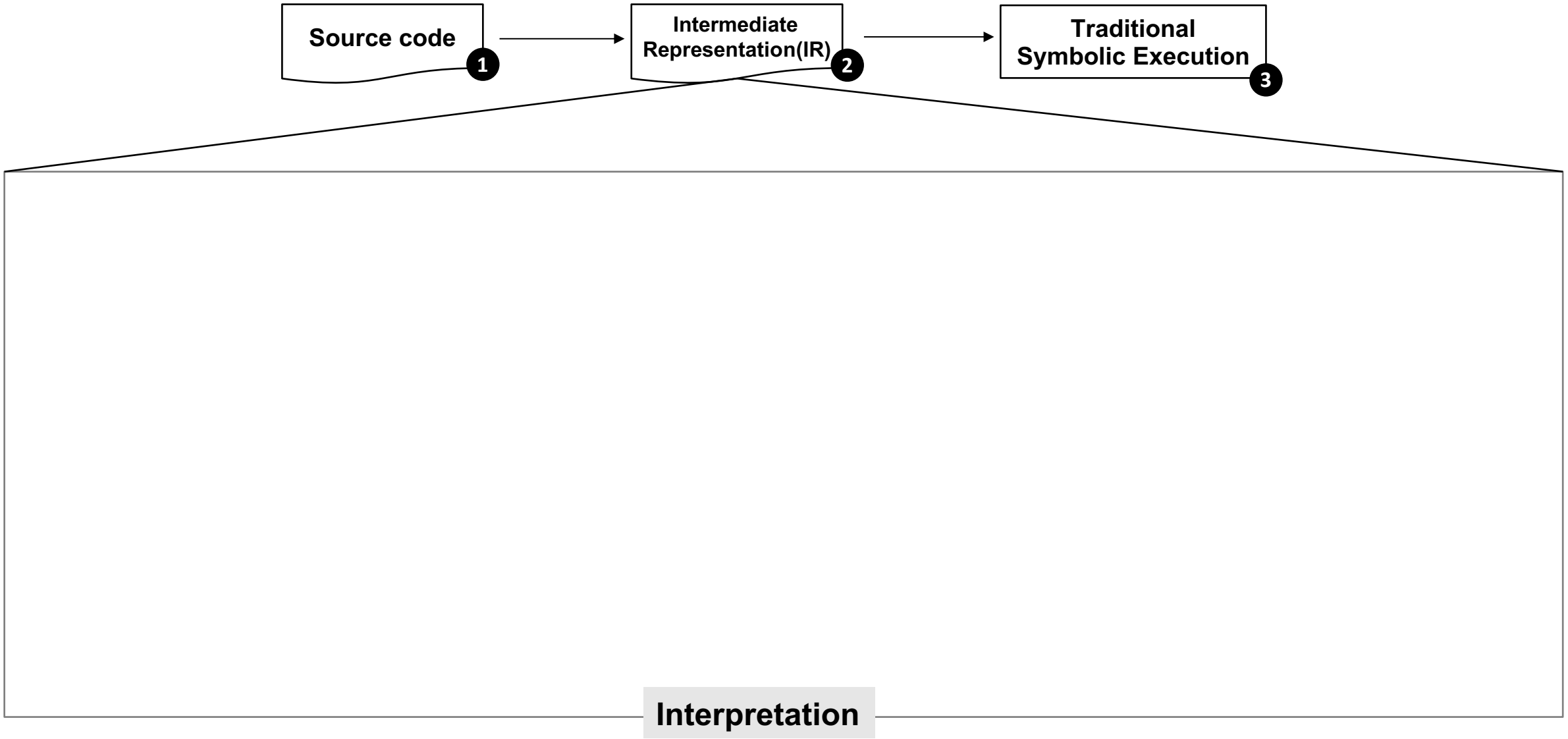
Efficiency **matters when performing symbolic execution**

Motivation

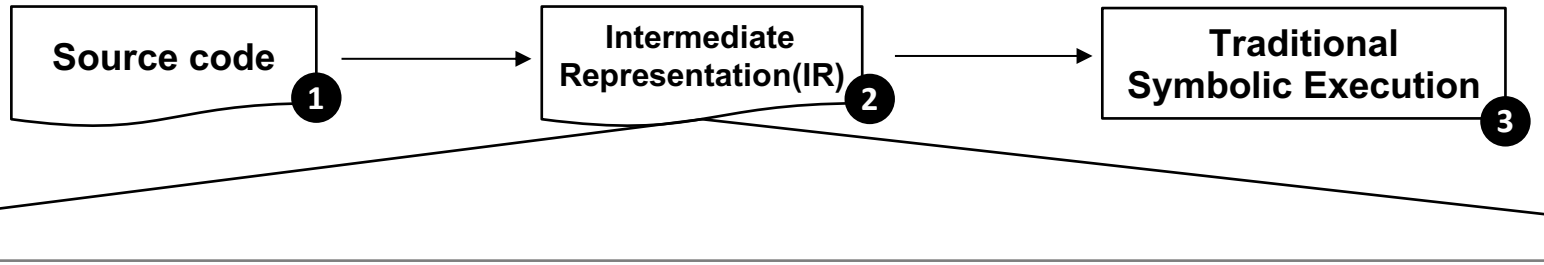
Motivation


Source code
1
Intermediate Representation(IR)
2
Traditional Symbolic Execution
3

Motivation


Source code
1
Intermediate Representation(IR)
2
Traditional Symbolic Execution
3
Interpretation

Motivation



(1) Observation

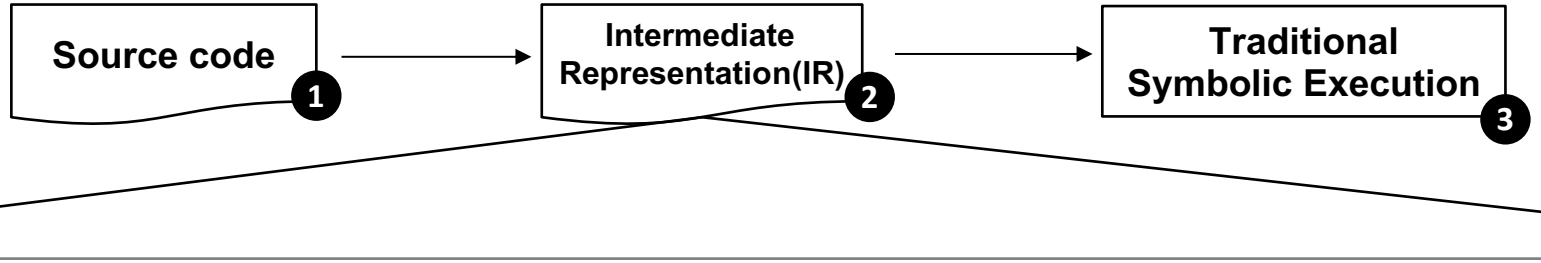
- The number of interpreted instructions tends to be **huge** (several billion only in one hour run)

```
Elapsed: 01:00:04
KLEE: done: explored paths = 125017
KLEE: done: avg. constructs per query = 74
KLEE: done: total queries = 8859
KLEE: done: valid queries = 6226
KLEE: done: invalid queries = 2633
KLEE: done: query cex = 8859

KLEE: done: total instructions = 605113213
KLEE: done: completed paths = 125017
KLEE: done: generated tests = 65
```

Interpretation

Motivation



(1) Observation

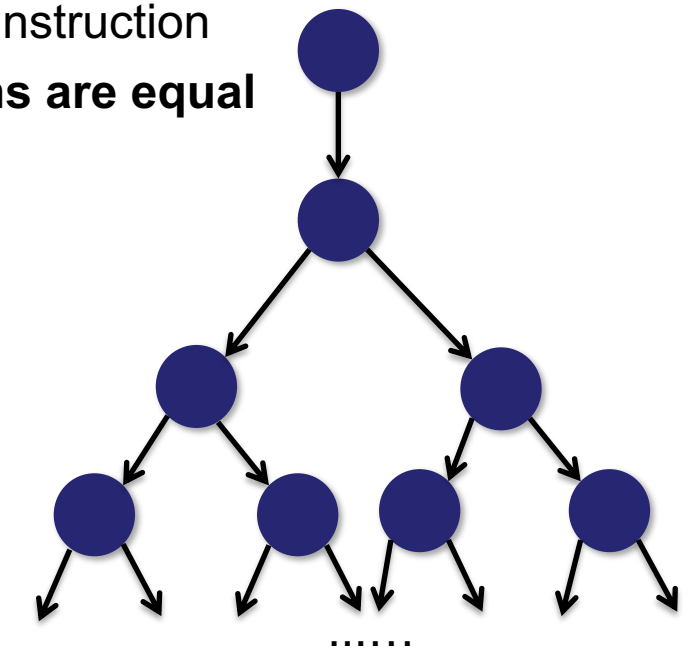
- The number of interpreted instructions tends to be **huge** (several billion only in one hour run)

```
Elapsed: 01:00:04
KLEE: done: explored paths = 125017
KLEE: done: avg. constructs per query = 74
KLEE: done: total queries = 8859
KLEE: done: valid queries = 6226
KLEE: done: invalid queries = 2633
KLEE: done: query cex = 8859

KLEE: done: total instructions = 605113213
KLEE: done: completed paths = 125017
KLEE: done: generated tests = 65
```

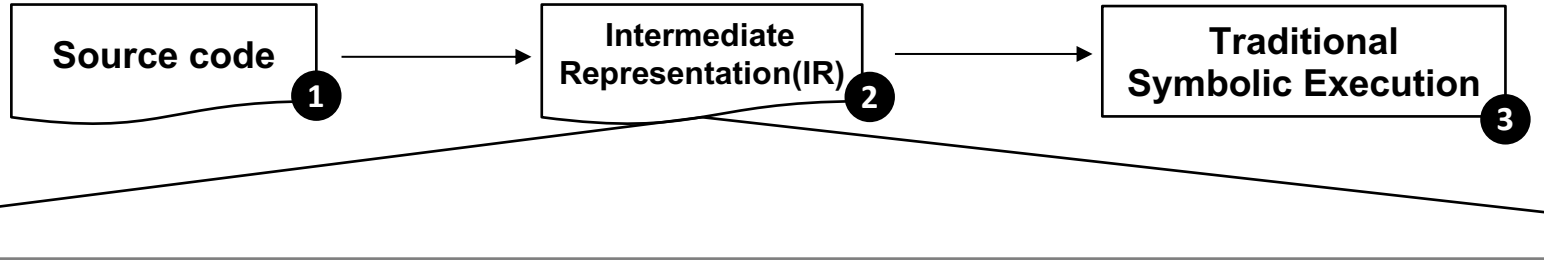
(2) Overheads in current symbolic execution

- The color depth represents the overheads of an interpreted instruction
- **All instructions are equal**



Interpretation

Motivation



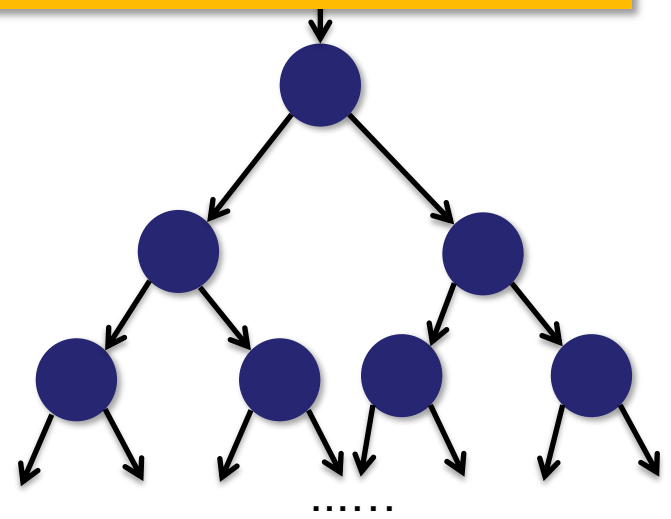
(1) Observation

(2) Overheads in current symbolic execution

Can we reduce the overhead of interpreted instructions for faster symbolic execution?

```
Elapsed: 01:00:04
KLEE: done: explored paths = 125017
KLEE: done: avg. constructs per query = 74
KLEE: done: total queries = 8859
KLEE: done: valid queries = 6226
KLEE: done: invalid queries = 2633
KLEE: done: query cex = 8859

KLEE: done: total instructions = 605113213
KLEE: done: completed paths = 125017
KLEE: done: generated tests = 65
```



Interpretation

Solution – FastKLEE (1/2)

[1] Ccured: George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak, and Westley Weimer. 2005. **CCured: type-safe retrofitting of legacy software**. ACM Trans. Program. Lang. Syst. 27, 3 (May 2005), 477–526.

Solution – FastKLEE (1/2)

- **Key insights**

[1] Ccured: George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak, and Westley Weimer. 2005. **CCured: type-safe retrofitting of legacy software**. ACM Trans. Program. Lang. Syst. 27, 3 (May 2005), 477–526.

Solution – FastKLEE (1/2)

SMU SINGAPORE MANAGEMENT UNIVERSITY | School of Information Systems

- **Key insights**
  - Only a small portion of memory-related instructions need bound checking

[1] Ccured: George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak, and Westley Weimer. 2005. **CCured: type-safe retrofitting of legacy software**. ACM Trans. Program. Lang. Syst. 27, 3 (May 2005), 477–526.

Solution – FastKLEE (1/2)

- **Key insights**
  - Only a small portion of memory-related instructions need bound checking
  - Reduce interpreting overhead of most frequently interpreted (i.e., load/store)

[1] Ccured: George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak, and Westley Weimer. 2005. **CCured: type-safe retrofitting of legacy software**. ACM Trans. Program. Lang. Syst. 27, 3 (May 2005), 477–526.

Solution – FastKLEE (1/2)

SMU SINGAPORE MANAGEMENT UNIVERSITY | School of Information Systems

- **Key insights**
  - Only a small portion of memory-related instructions need bound checking
  - Reduce interpreting overhead of most frequently interpreted (i.e., load/store)

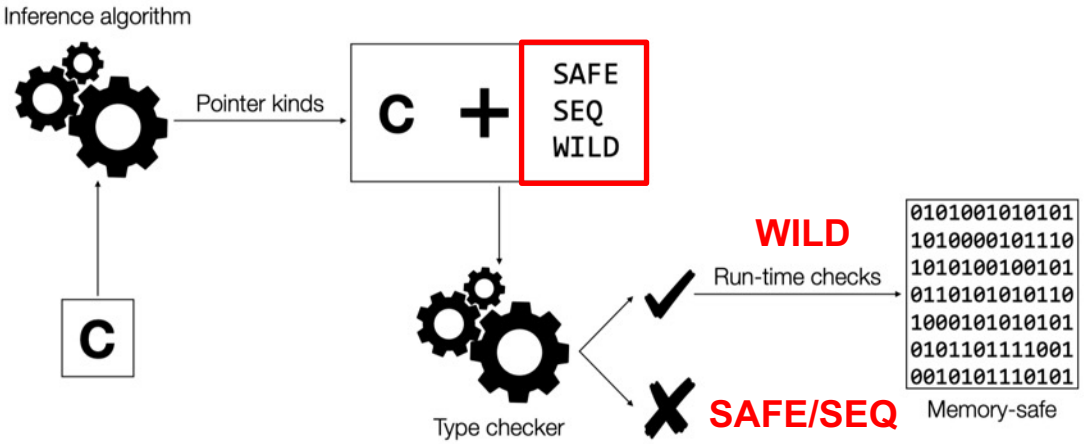
  - Inspired by ***Type Inference*** system [1]

[1] Ccured: George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak, and Westley Weimer. 2005. **CCured: type-safe retrofitting of legacy software**. ACM Trans. Program. Lang. Syst. 27, 3 (May 2005), 477–526.

Solution – FastKLEE (1/2)

- **Key insights**
  - Only a small portion of memory-related instructions need bound checking
  - Reduce interpreting overhead of most frequently interpreted (i.e., load/store)

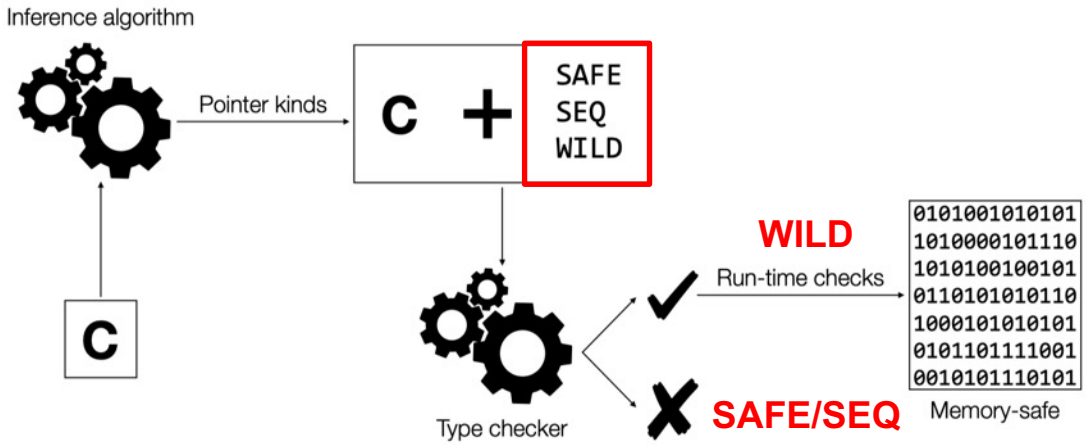
  - Inspired by ***Type Inference*** system [1]



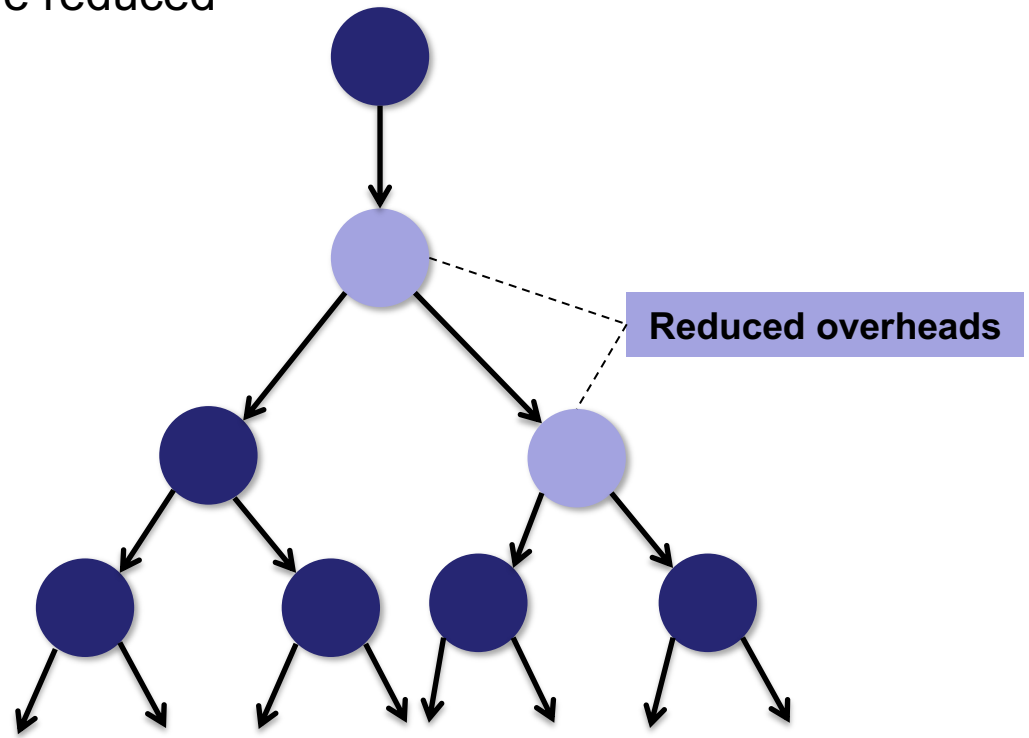
[1] Ccured: George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak, and Westley Weimer. 2005. **CCured: type-safe retrofitting of legacy software**. ACM Trans. Program. Lang. Syst. 27, 3 (May 2005), 477–526.

Solution – FastKLEE (1/2)

- **Key insights**
  - Only a small portion of memory-related instructions need bound checking
  - Reduce interpreting overhead of most frequently interpreted (i.e., load/store)
  - Inspired by ***Type Inference*** system [1]



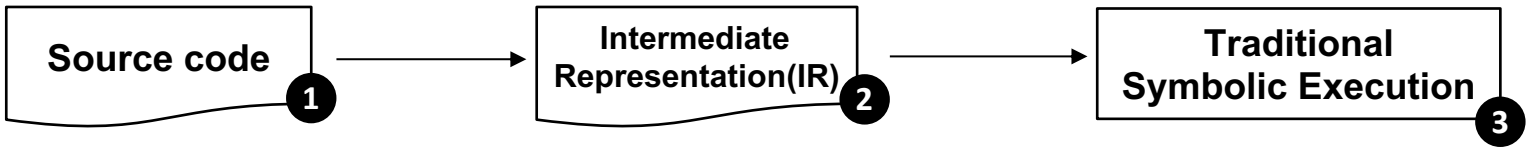
- **Advantage: overheads in FastKLEE**
  - Interpretation overheads for some instructions are reduced



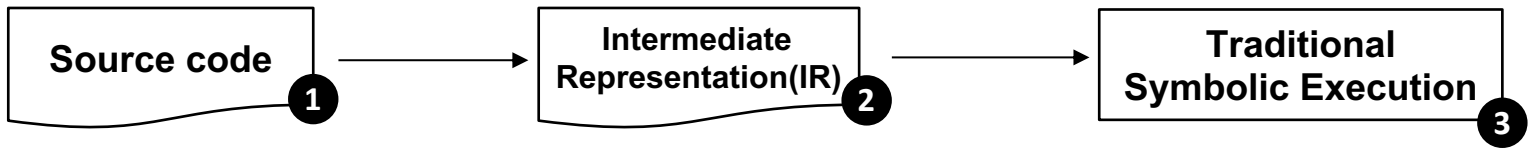
[1] Ccured: George C. Necula, Jeremy Condit, Matthew Harren, Scott McPeak, and Westley Weimer. 2005. **CCured: type-safe retrofitting of legacy software**. ACM Trans. Program. Lang. Syst. 27, 3 (May 2005), 477–526.

Solution – FastKLEE (2/2)

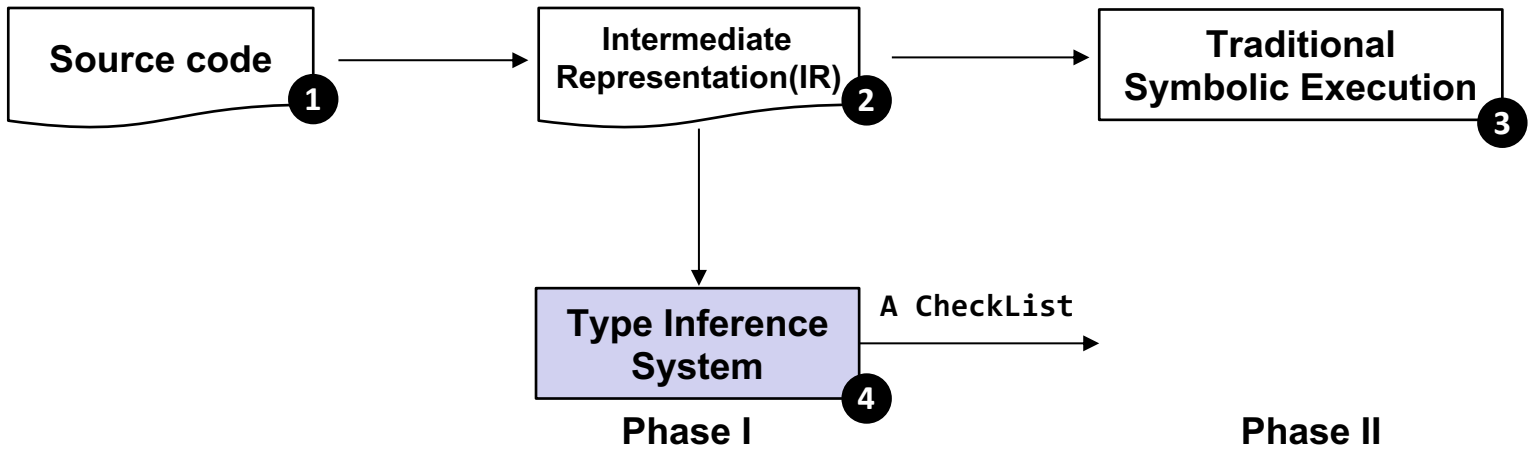
Solution – FastKLEE (2/2)



Solution – FastKLEE (2/2)

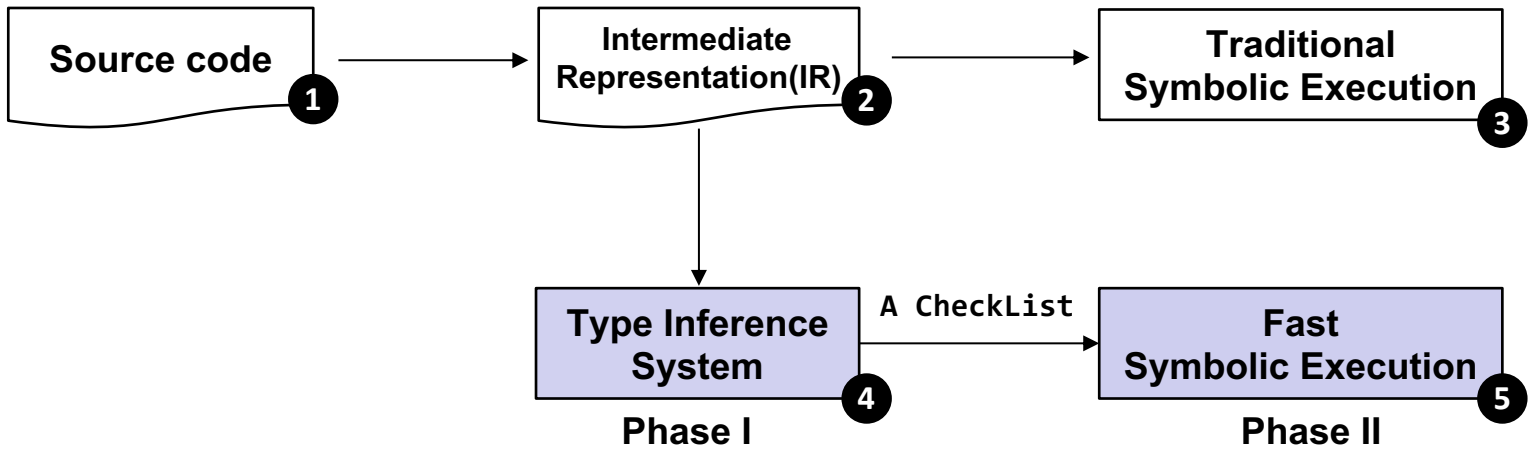


Solution – FastKLEE (2/2)



4 • **Phase I**: Introduce a **Type Inference System** to classify memory-related instruction types
  – ***Unsafe*** memory instructions will be stored in **CheckList**

Solution – FastKLEE (2/2)



4 • **Phase I**: Introduce a **Type Inference System** to classify memory-related instruction types
  – ***Unsafe*** memory instructions will be stored in **CheckList**

5 • **Phase II**: Conduct **Customized Memory Operation** in Fast symbolic execution
  – Only perform checking for ***Unsafe*** memory instructions during interpretation

Preliminary Evaluation

Preliminary Evaluation

SMU SINGAPORE MANAGEMENT UNIVERSITY | School of Information Systems

- **Benchmark**
  - GNU Coreutils
  - ~ 1-5k SLOC for each test program

Preliminary Evaluation

SMU SINGAPORE MANAGEMENT UNIVERSITY | School of Information Systems

- **Benchmark**
  - GNU Coreutils
  - ~ 1-5k SLOC for each test program

- **Metric**
  - ***Speedups***: the **time** spent on exploring the same number of instructions

$$Speedups: \frac{T_{baseline} - T_{our}}{T_{baseline}} \times 100$$

- $T_{baseline}$ **: existing approach**
- T_{our} **: our approach**

Preliminary Evaluation

- **Benchmark**
  - GNU Coreutils
  - ~ 1-5k SLOC for each test program

- **Metric**
  - ***Speedups***: the **time** spent on exploring the same number of instructions

$$Speedups: \frac{T_{baseline} - T_{our}}{T_{baseline}} \times 100$$

- $T_{baseline}$: **existing approach**
- T_{our} : **our approach**

- **Results**

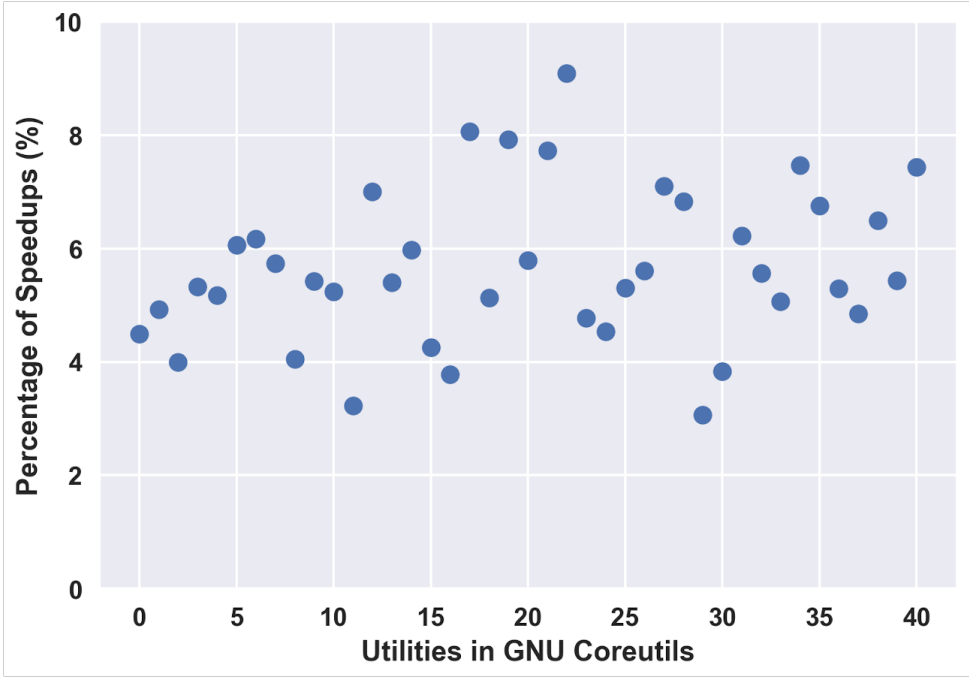


Fig. Scatter plot of the improvement in speedups

Preliminary Evaluation

- **Benchmark**
  - GNU Coreutils
  - ~ 1-5k SLOC for each test program

- **Metric**
  - ***Speedups***: the **time** spent on exploring the same number of instructions

$$Speedups: \frac{T_{baseline} - T_{our}}{T_{baseline}} \times 100$$

- $T_{baseline}$ **: existing approach**
- T_{our} **: our approach**

- **Results**

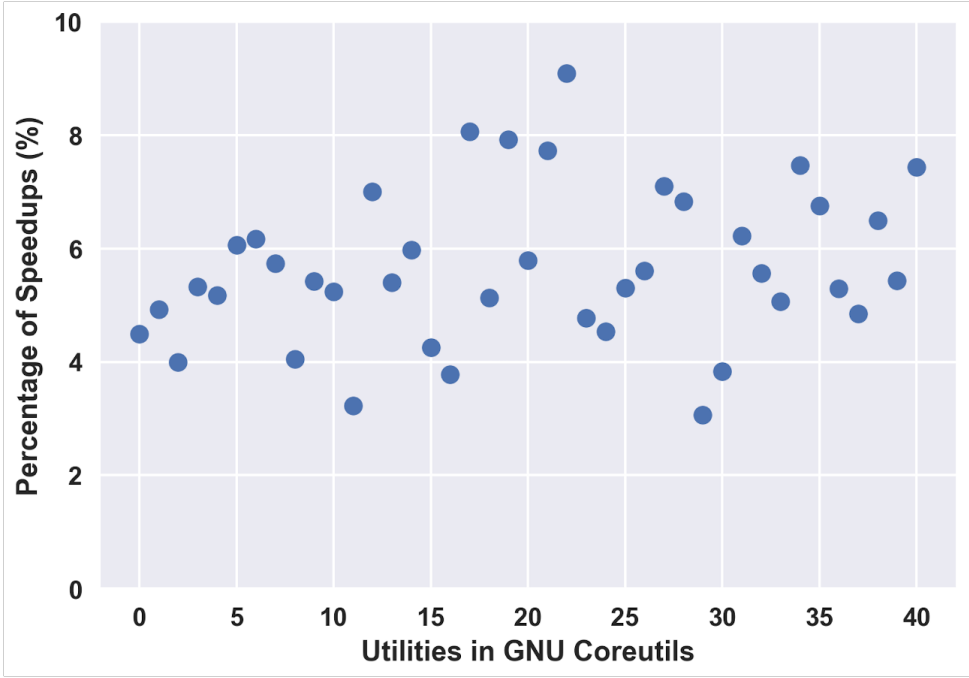


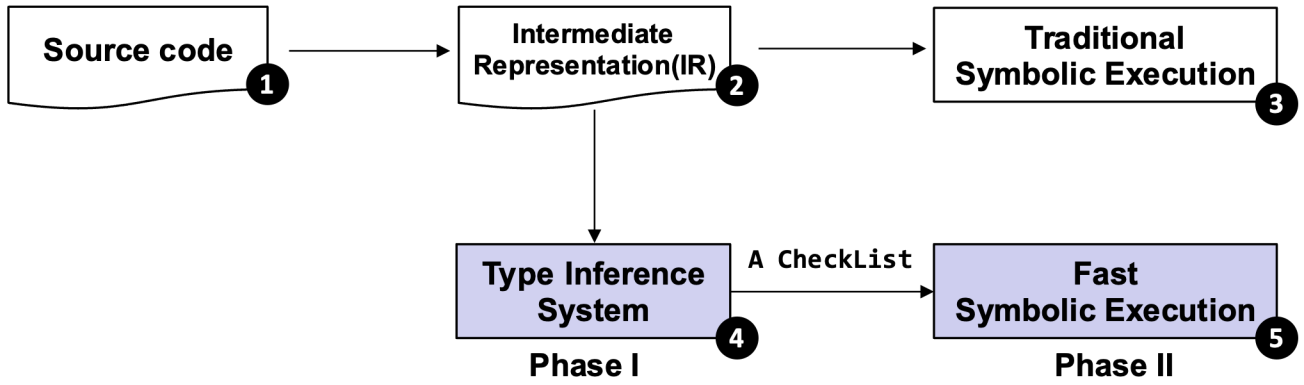
Fig. Scatter plot of the improvement in speedups

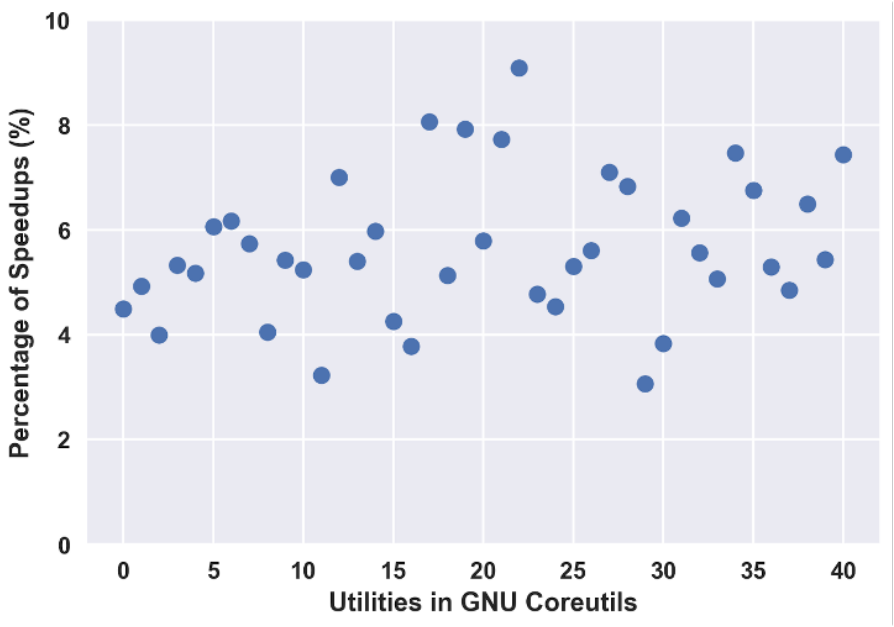
  - FastKLEE can reduce by up to **9.1%** time compared with the state-of-the-art approach (i.e., KLEE)

Conclusion

Conclusion

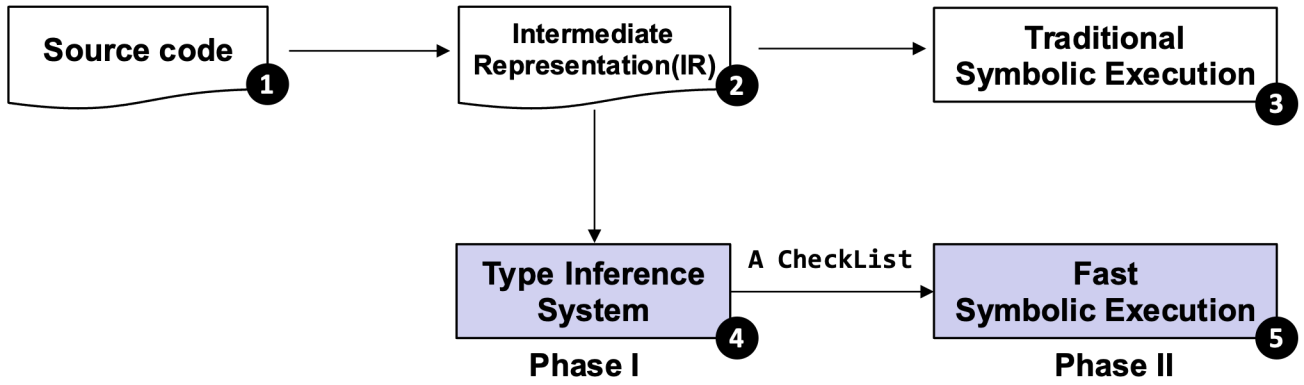
- We present **FastKLEE**, a faster symbolic execution via reducing the interpretation overheads

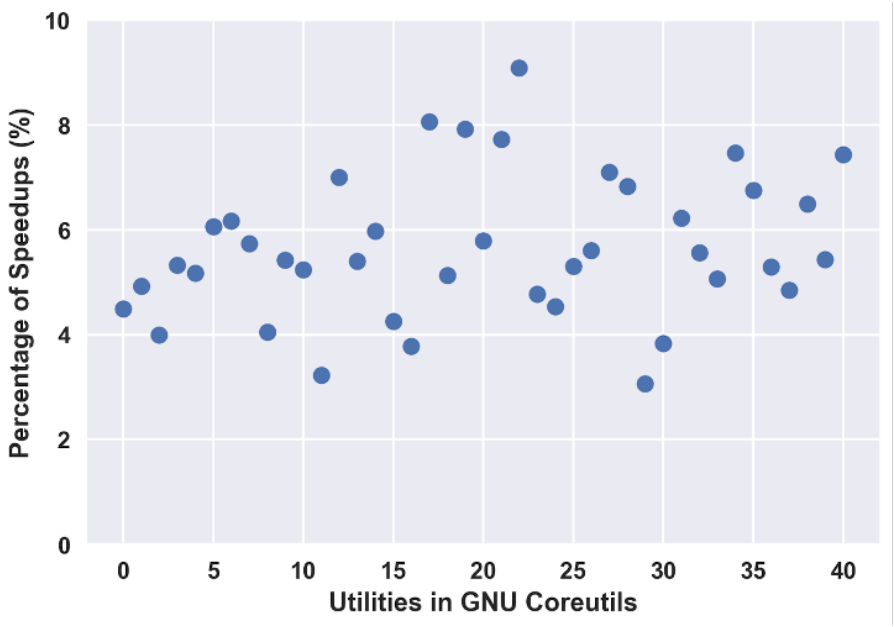




Conclusion

- We present **FastKLEE**, a faster symbolic execution via reducing the interpretation overheads





- **Future work**
  - Follow the idea of FastKLEE to conduct **vulnerability-oriented** path exploration for symbolic execution
    - Valuable paths: more likely to contain vulnerabilities

Code

Video Demo

Thank you and Questions !

FastKLEE: Faster Symbolic Execution via Reducing Redundant Bound Checking of Type-Safe Pointers

Haoxin Tu, Lingxiao Jiang, Xuhua Ding (Singapore Management University)
He Jiang (Dalian University of Technology)