

Coverage-Guided *Tensor Compiler Fuzzing* with Joint IR-Pass Mutation

Jiawei Liu, Yuxiang Wei, Sen Yang, Yinlin Deng, Lingming Zhang

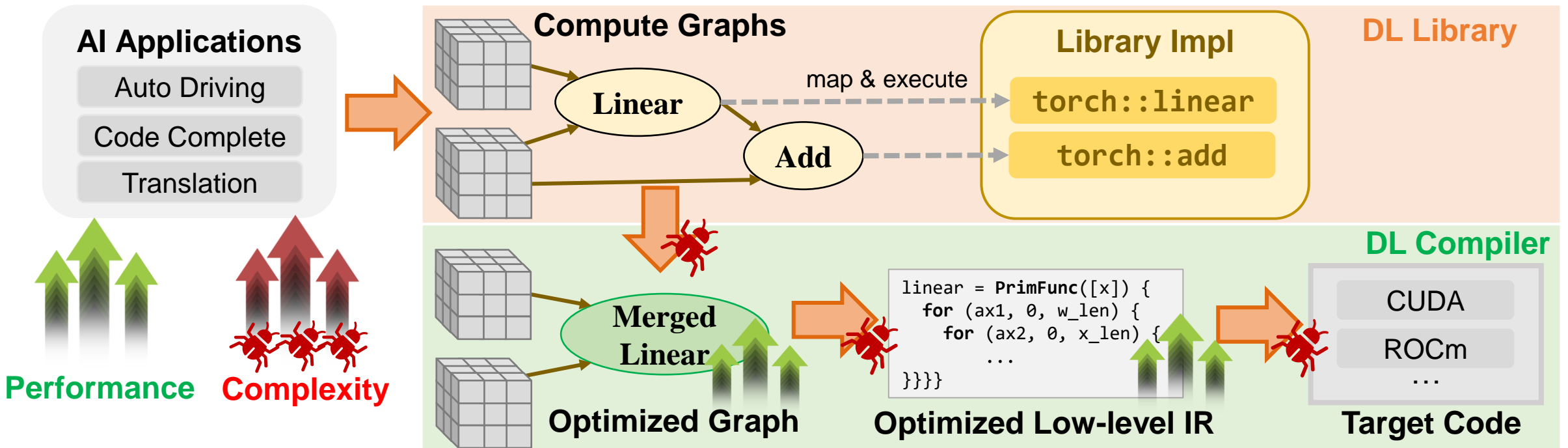


jiawei6@illinois.edu



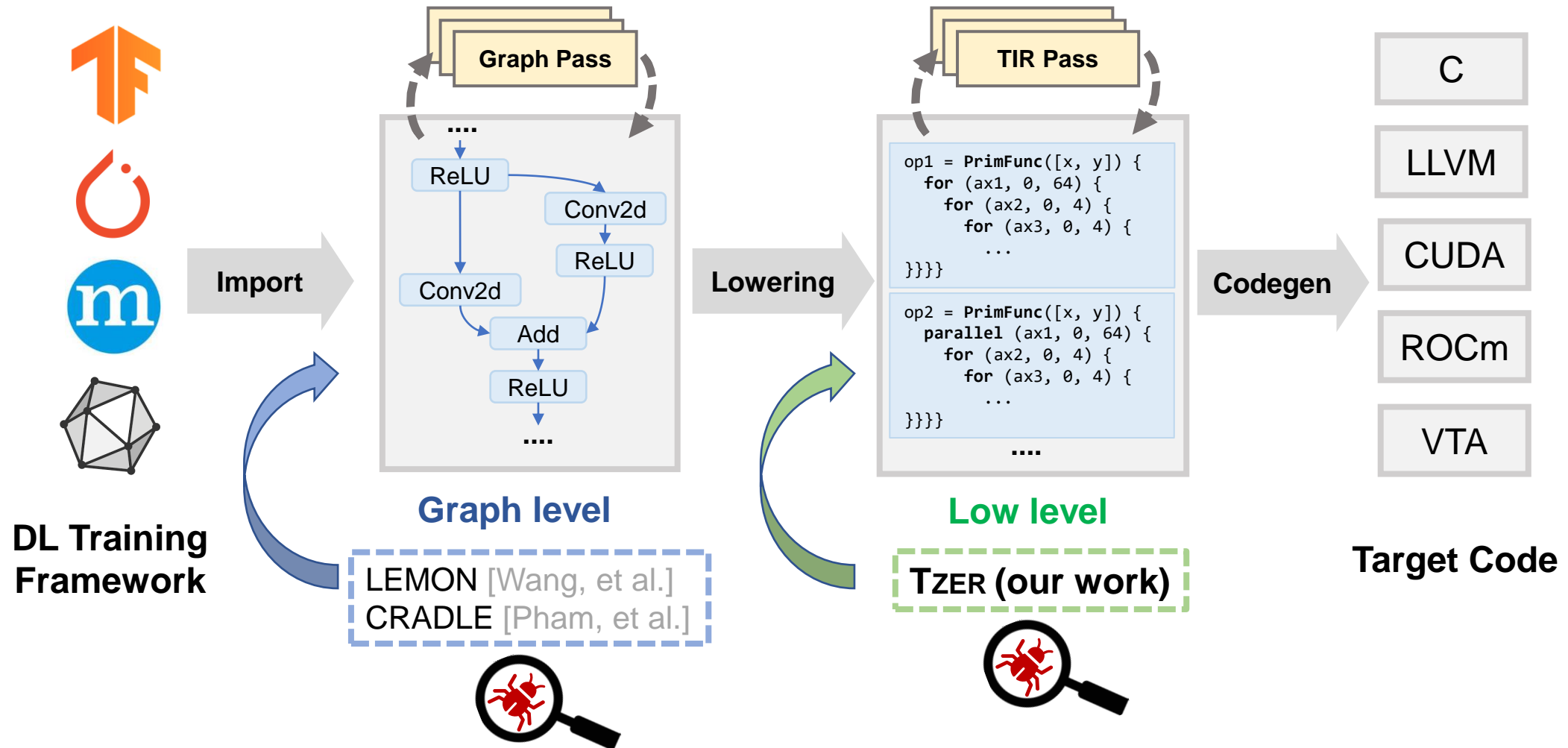
DL Frameworks: from Libraries to Compilers

- ❖ **First** gen.: **Libraries**, e.g., PyTorch, TensorFlow.
- ❖ **Second** gen.: **Compilers**, e.g., TVM, TensorFlow XLA, PyTorch JIT.

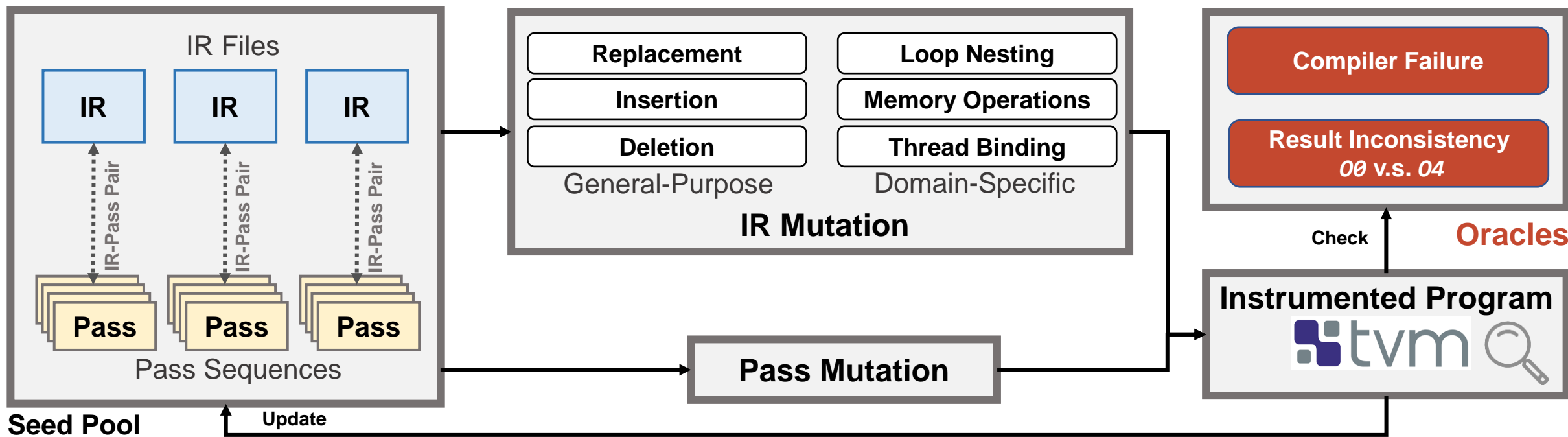
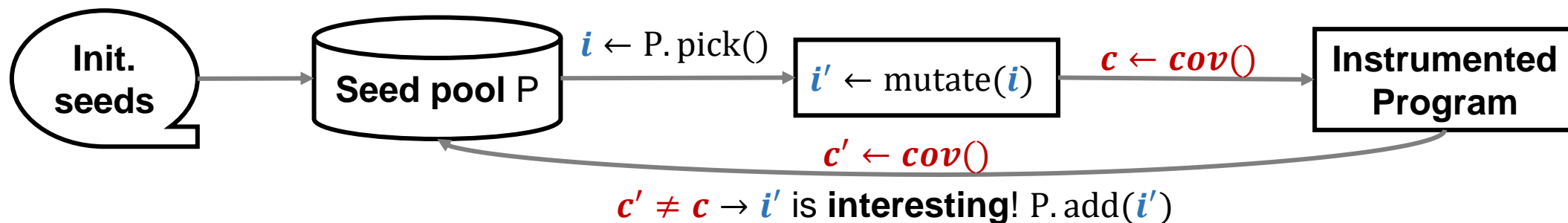


Goal: automatically detect bugs in tensor compilers.

A Typical Workflow of Tensor Compiler

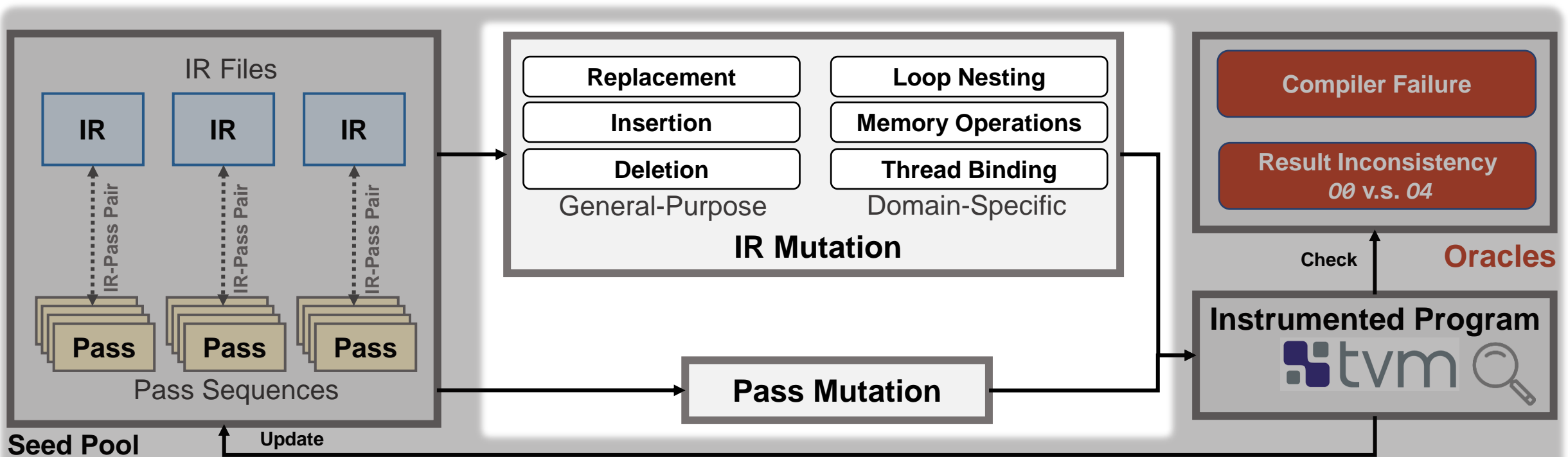


Fuzzing Loop in TZER



Generating New Test-Cases with Mutation

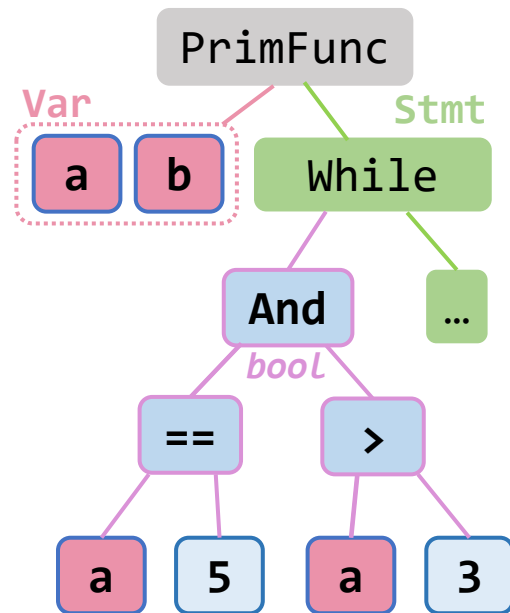
- ❖ **IR Mutation:** practice the compilation of various IR structures.
- ❖ **Pass Mutation:** practice the compatibility of pass orders.



General IR Mutation

```
PrimFunc(a, b) {  
  while (a == 5 && a > 3) {  
    ...  
  }  
}
```

- ❖ Select target AST node to mutate.
- ❖ Perform insertion/deletion/replacement with constraints.
 - e.g., accessible variables, type requirements, etc.

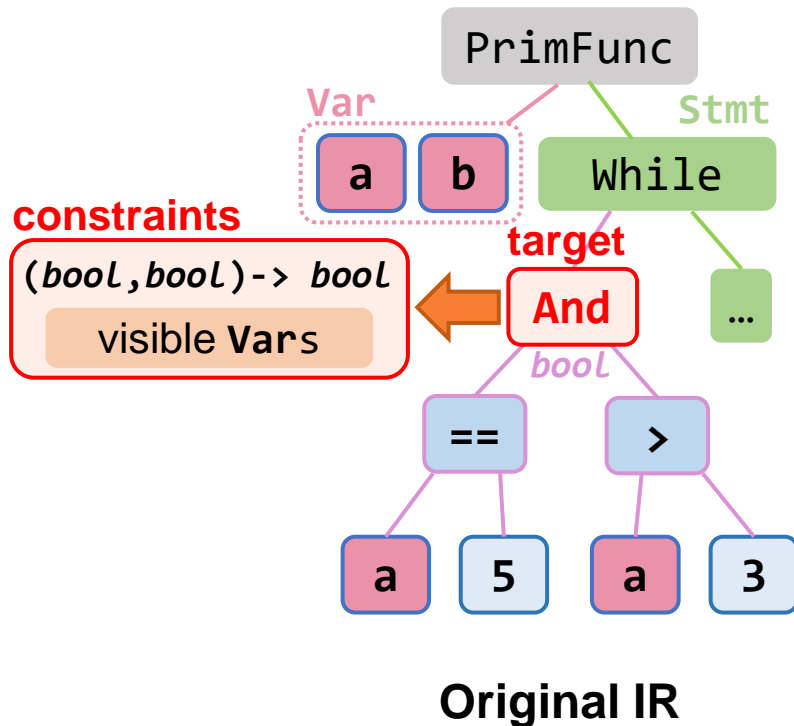


Original IR

General IR Mutation

```
PrimFunc(a, b) {  
  while (a == 5 && a > 3) {  
    ...  
  }  
}
```

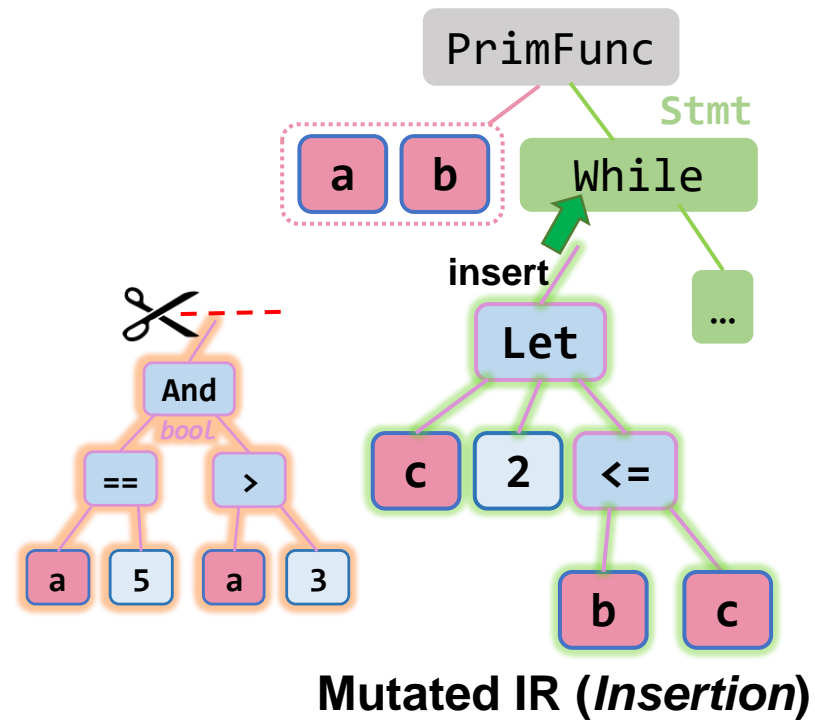
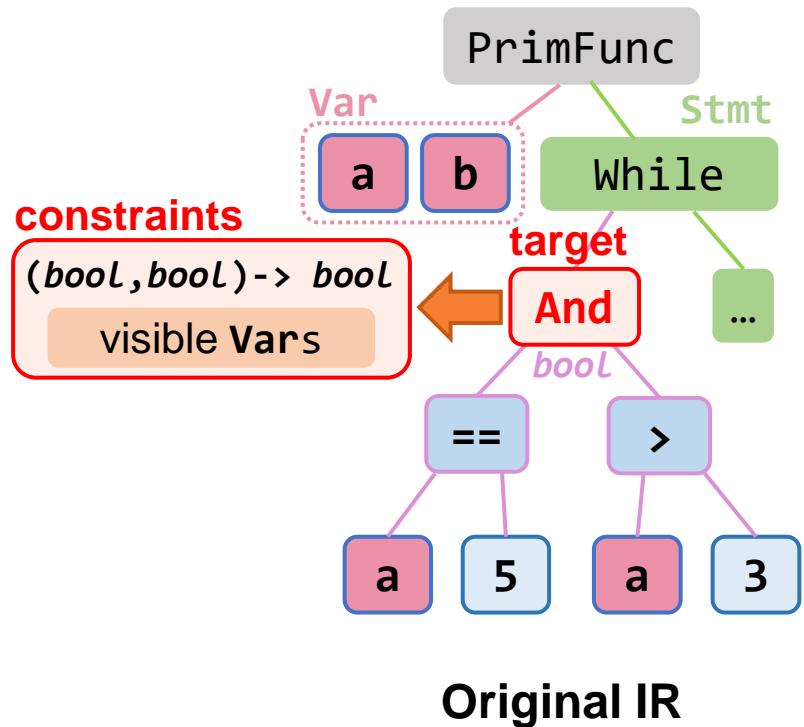
- ❖ Select **target AST node** to mutate.
- ❖ Perform insertion/deletion/replacement with **constraints**.
 - e.g., accessible variables, type requirements, etc.



General IR Mutation

```
PrimFunc(a, b) {  
- while (a == 5 && a > 3) {  
+ while (c = 2; b <= c ) {  
  ...  
}  
}
```

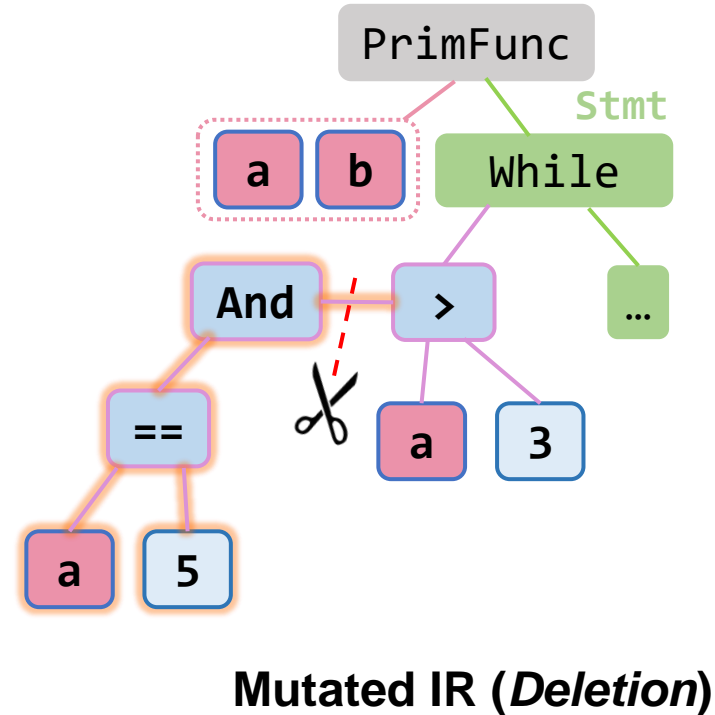
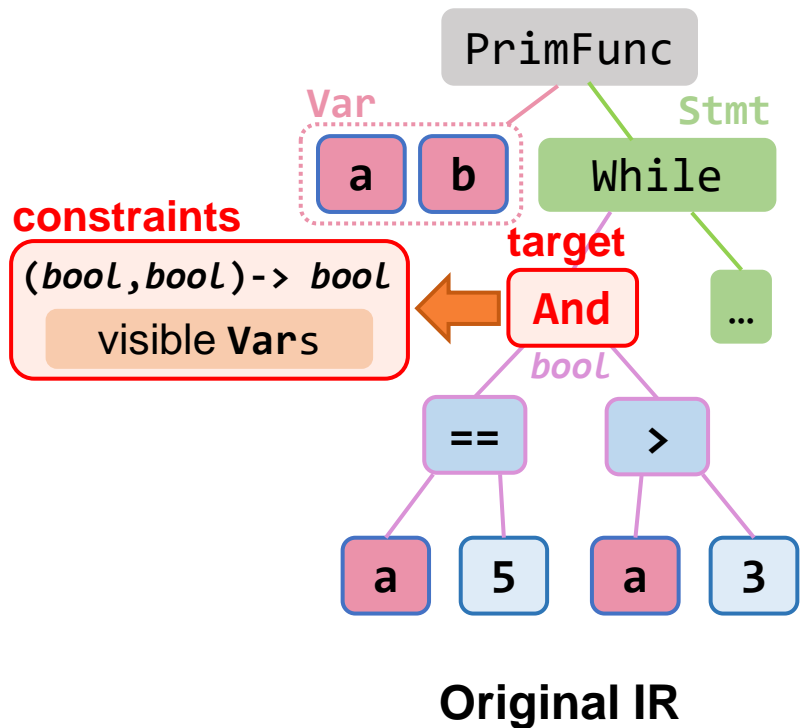
- ❖ Select **target AST node** to mutate.
- ❖ Perform **insertion**/deletion/replacement with **constraints**.
 - e.g., accessible variables, type requirements, etc.



General IR Mutation

```
PrimFunc(a, b) {  
- while (a == 5 && a > 3) {  
+ while (a > 3) {  
  ...  
}  
}
```

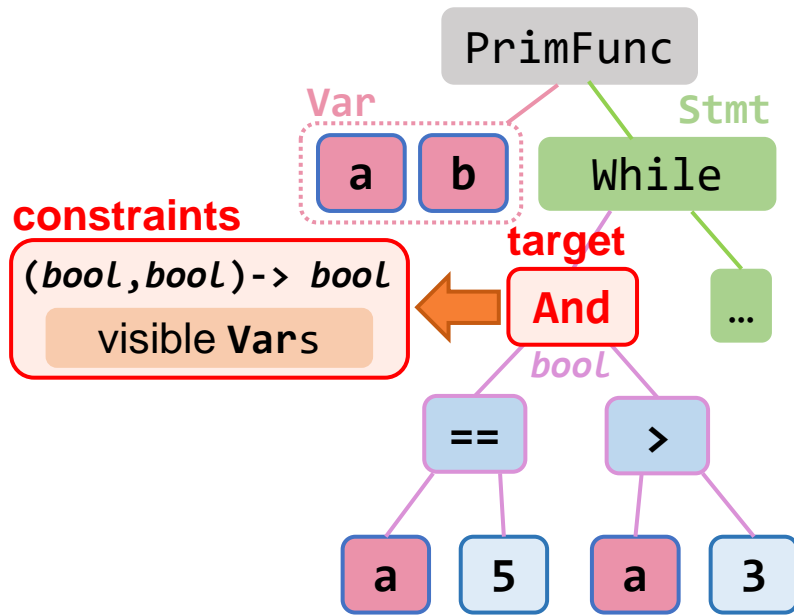
- ❖ Select **target AST node** to mutate.
- ❖ Perform insertion/**deletion**/replacement with **constraints**.
 - e.g., accessible variables, type requirements, etc.



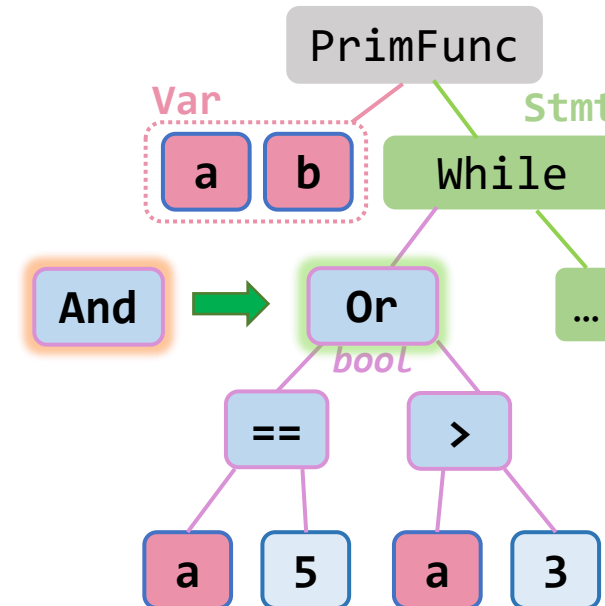
General IR Mutation

- ❖ Select **target AST node** to mutate.
- ❖ Perform insertion/deletion/**replacement** with **constraints**.
 - e.g., accessible variables, type requirements, etc.

```
PrimFunc(a, b) {  
- while (a == 5 && a > 3) {  
+ while (a == 5 || a > 3) {  
  ...  
}  
}
```



Original IR



Mutated IR (Node Replacement)

Domain-specific IR Mutation

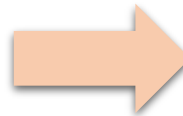
- ❖ Tensor program optimization is highly related to
 - ❖ **Loops**: add outer loops;
 - ❖ **Memory**: mutate memory load/store;
 - ❖ **Threads**: mutate threading patterns.

Domain-specific IR Mutation

- ❖ Tensor program optimization is highly related to
 - ❖ **Loops**: insert loops;
 - ❖ **Memory**: mutate memory load/store;
 - ❖ **Threads**: mutate threading patterns.

```
PrimFunc(...) {  
  buf[0] = 1  
}
```

Loop Nesting



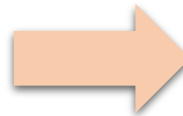
```
PrimFunc(...) {  
  // attr ... unroll_max_step = 2  
  unrolled (i, 0, 16) {  
    unrolled (j, 0, 16) {  
      buf[0] = 1  
    }  
  }  
}
```

Domain-specific IR Mutation

- ❖ Tensor program optimization is highly related to
 - ❖ **Loops**: add outer loops;
 - ❖ **Memory**: mutate memory load/store;
 - ❖ **Threads**: mutate threading patterns.

```
PrimFunc(...) {  
  // attr ... unroll_max_step = 2  
  unrolled (i, 0, 16) {  
    unrolled (j, 0, 16) {  
      buf[0] = 1  
    }  
  }  
}
```

Memory Operation



```
PrimFunc(...) {  
  // attr ... unroll_max_step = 2  
  unrolled (i, 0, 16) {  
    unrolled (j, 0, 16) {  
      buf[i * 16 + j] = buf[i + j * 16]  
    }  
  }  
}
```

Domain-specific IR Mutation

- ❖ Tensor program optimization is highly related to
 - ❖ **Loops**: add outer loops;
 - ❖ **Memory**: mutate memory load/store;
 - ❖ **Threads**: mutate threading patterns.

```
PrimFunc(...) {  
  // attr ... unroll_max_step = 2  
  unrolled (i, 0, 16) {  
    unrolled (j, 0, 16) {  
      buf[i * 16 + j] = buf[i + j * 16]  
    }  
  }  
}
```

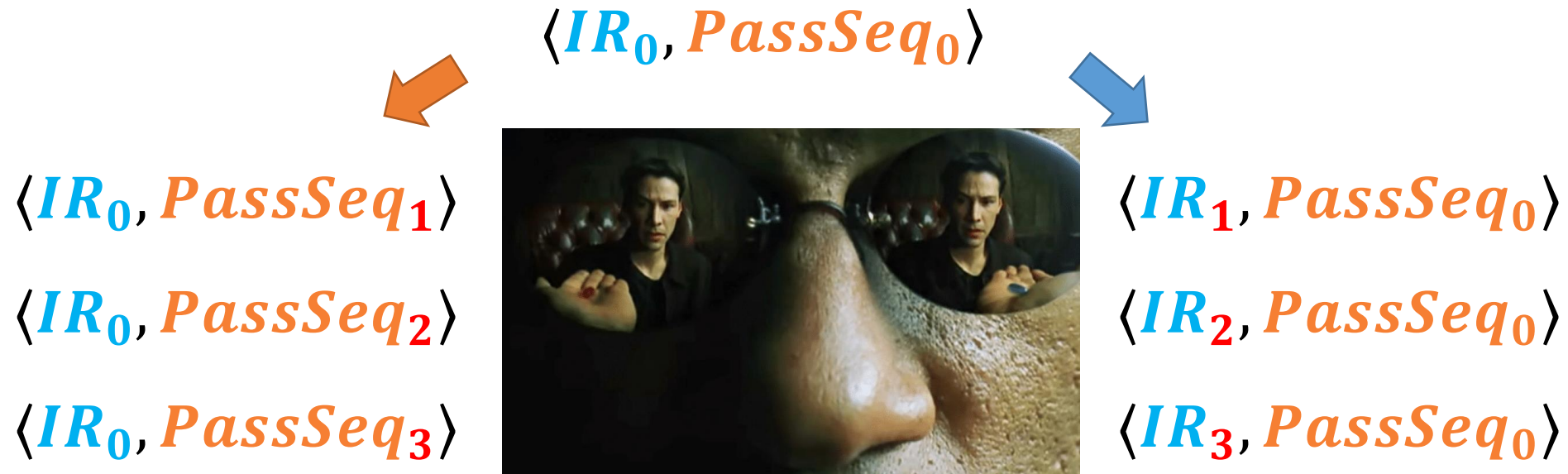
Thread Binding



```
PrimFunc([]) {  
  // attr ... virtual_thread = 2  
  launch_thread (t, 0, 2) {  
    // attr ... unroll_max_step = 2  
    unrolled (i, 0, 16) {  
      unrolled (j, 0, 16) {  
        buf[i * 16 + j] = buf[i + j * 16]  
      }  
    }  
  }  
}
```

How to mutate $\langle IR, PassSeq \rangle$ jointly?

Coverage is sensitive to mutation frequency of **IR** & **PassSeq**.



Always mutate **PassSeqs**?

- ❖ Less coverage-efficient than IR mutation.
- ❖ Lower compile rate by specific pairs (bug duplicates).

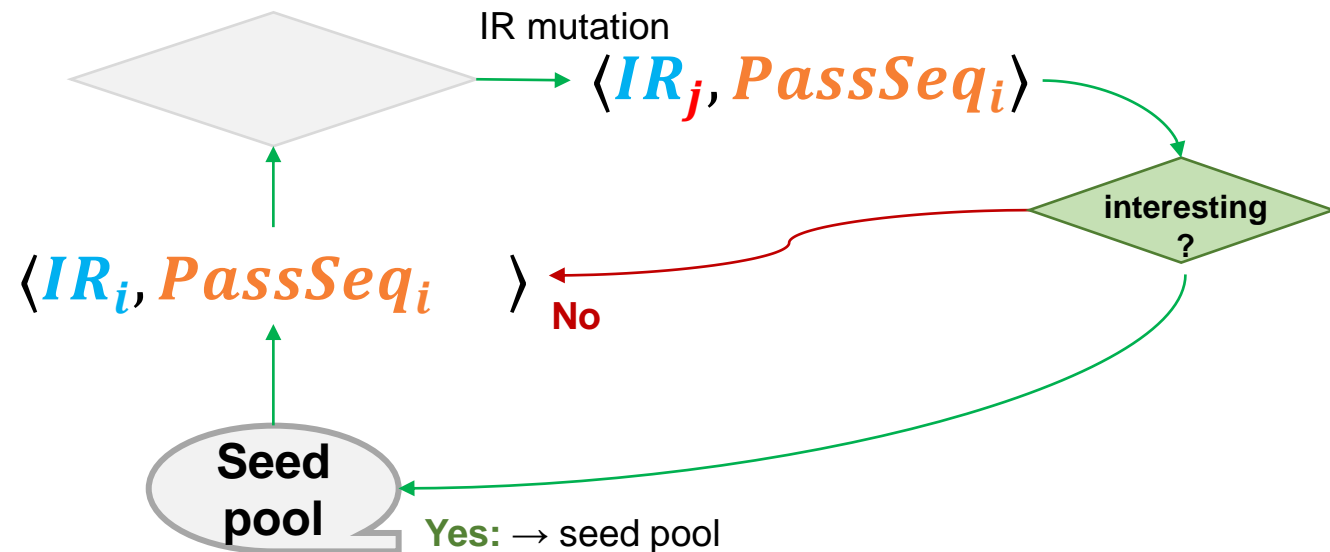
Always mutate **IRs**?

- ❖ Limited chances for practicing pass orders.

Joint IR-Pass Mutation

Solutions:

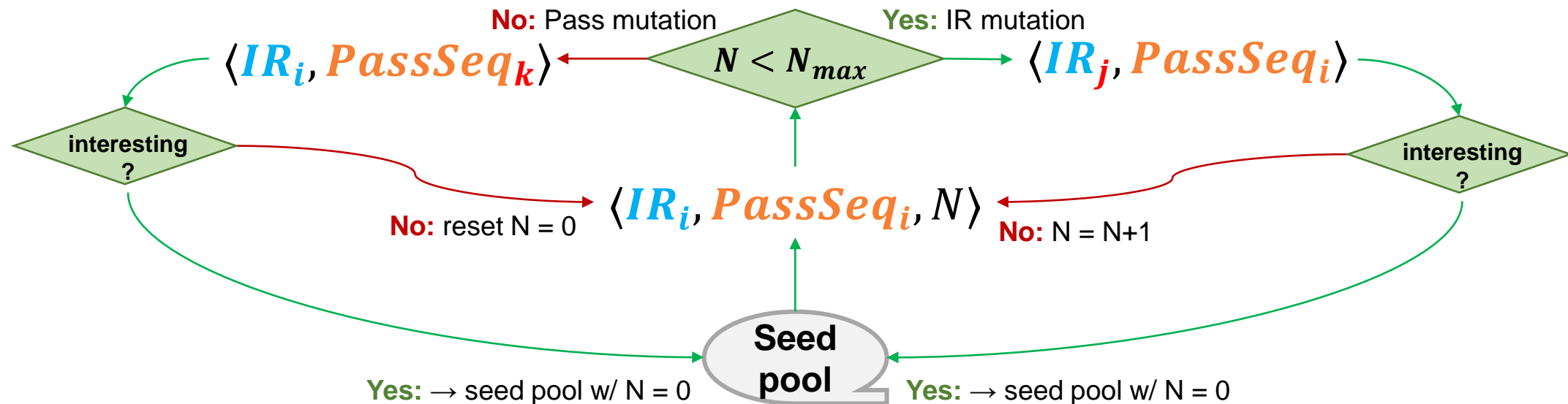
- ❖ Prefer IR mutation with interleaving factor N_{max}
- ❖ Avoid uninteresting $\langle IR, PassSeq \rangle$ (no new cov./uncompilable)



Joint IR-Pass Mutation

Solutions:

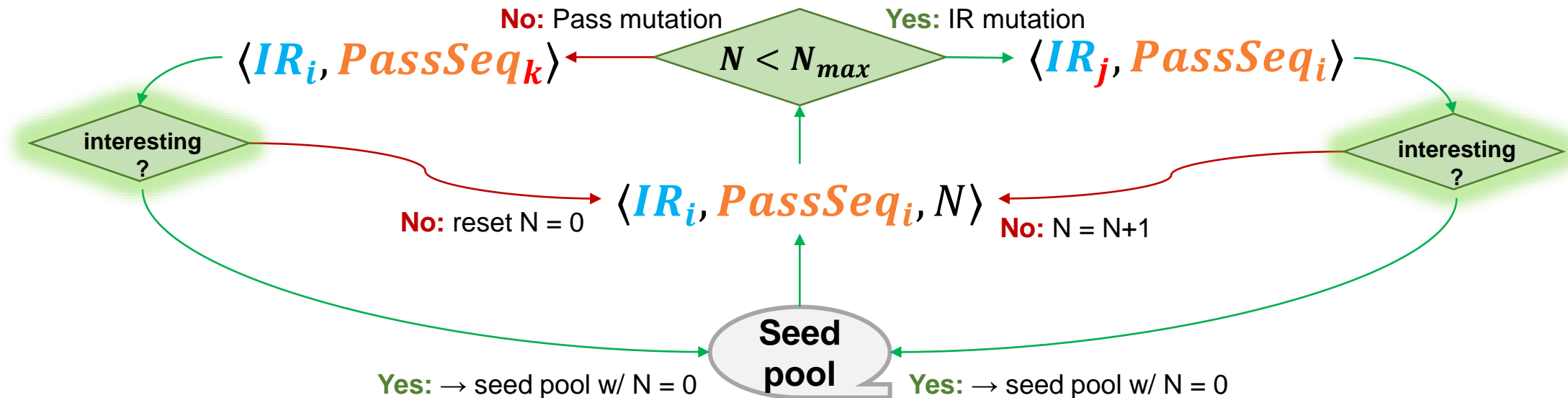
- ❖ Prefer IR mutation with interleaving factor N_{max}
- ❖ Avoid uninteresting $\langle IR, PassSeq \rangle$ (no new cov./uncompilable)



Joint IR-Pass Mutation

Solutions:

- ❖ Prefer IR mutation with interleaving factor N_{max}
- ❖ **Avoid uninteresting $\langle IR, PassSeq \rangle$ (no new cov./uncompilable)**



Bug Finding in TVM

49 reported; 37 confirmed; 25 fixed

Among all (37) confirmed bugs:

❖ **LEMON: 3 (8%)**

Fuzzing at **Graph** Level

❖ **TVMFuzz: 6 (16%)**

Fuzzing at **TIR** Level

❖ **LibFuzzer: 3 (8%)**

Fuzzing at **Binary** Level

❖ **TZER without pass mutation: 17 (46%)**

Sample: Violation of IR immutability

- The IR module in TVM is copied-on-write.
- The `ToBasicBlockNormalForm` pass could write the input IR in place even if it is not uniquely owned.
- This is because the use of `T* operator->() const` which actually returns a (non-const) mutable.

```
- IRModule ToBasicBlockNormalForm(const IRModule& mod) {  
+ IRModule ToBasicBlockNormalForm(const IRModule& mod_) {  
+     auto mod = IRModule(mod_>functions, mod_>type_definitions, ...); // Deep copy.
```

[tvm/8778: \[BUG\] ToBasicBlockNormalForm immutability](#)

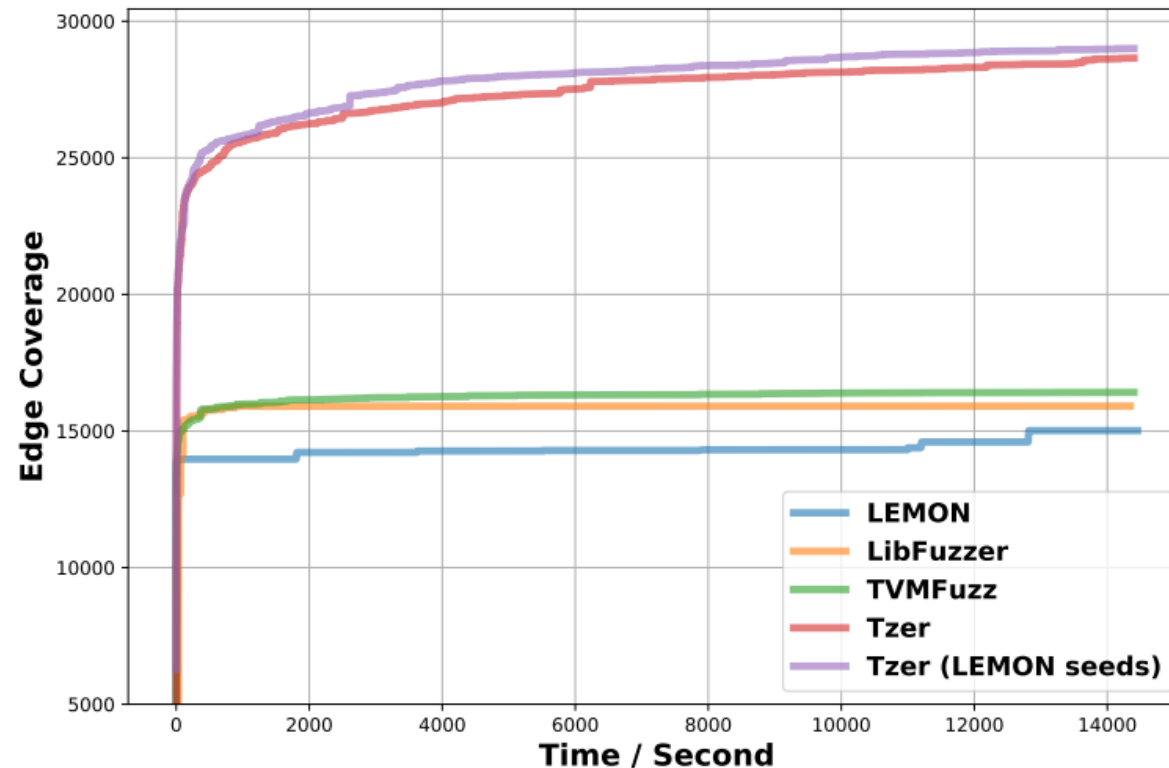
Sample: Out of Bound Read

- The **IRConvertSSA** pass can crash for out-of-bound read when any of the elements in **scope_** is an empty vector.

```
-   if (scope_.count(v)) {  
+   if (scope_.count(v) && !scope_[v].empty())  
       return Load(op->dtype, scope_[v].back(), op->index, op->predicate);
```

[tvm/8930: \[Bugfix\] Add check to avoid calling back\(\) on an empty container](#)

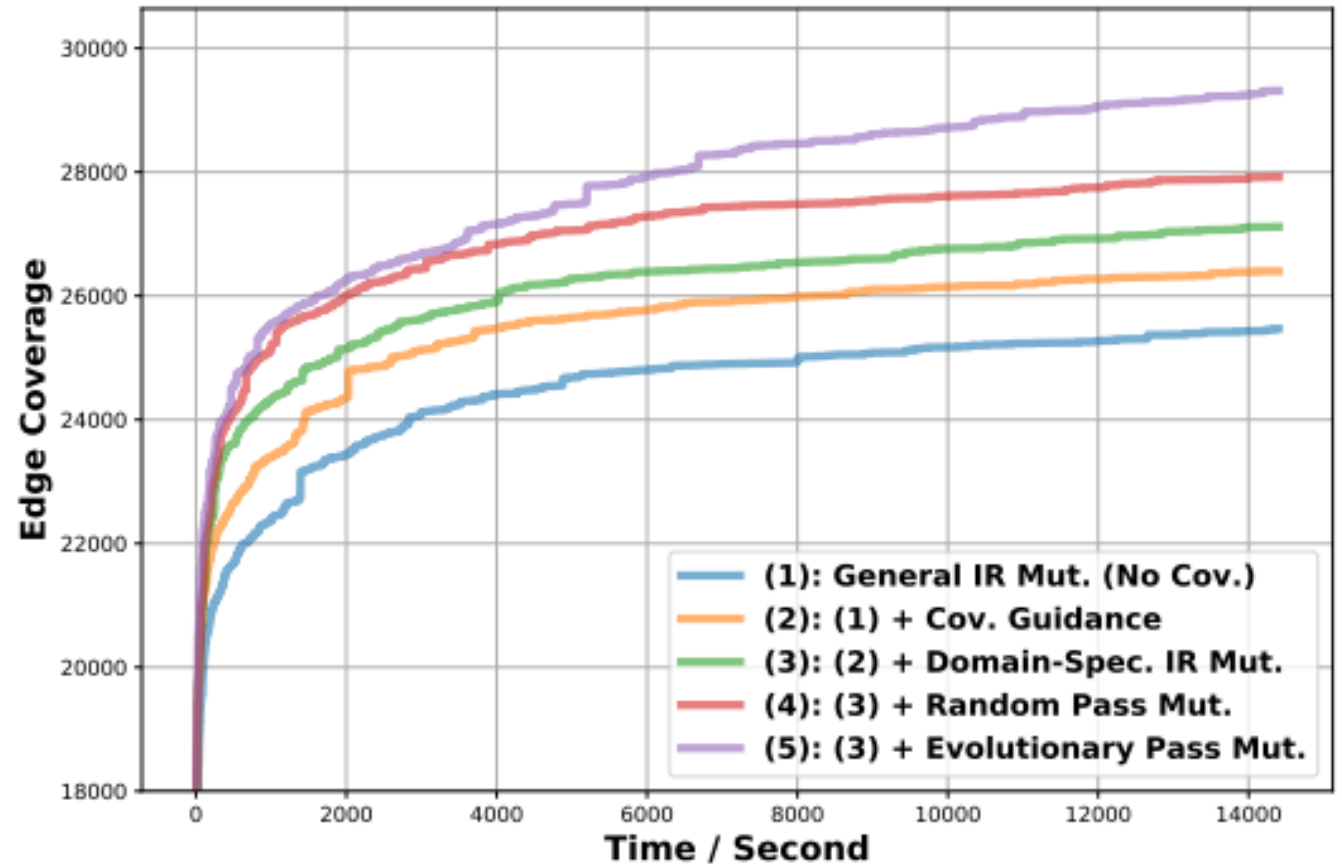
CFG Edge Coverage over 4 Hours



1.75x higher than TVMFuzz

Coverage: Ablation Study

- ✓ Coverage Guidance.
- ✓ Domain-spec. IR Mut.
- ✓ Pass Mutation.
- ✓ Evolutionary Fuzzing.



Summary

- ❖ **TZER: a tensor compiler fuzzer**
 - ❖ General & Domain-spec. IR Mutation
 - ❖ Pass Mutation
 - ❖ Coverage-guided Evolutionary Fuzzing
- ❖ **Found 49 bugs (37 confirmed)**

TZER
Tzer
documentation

Artifact Overview of Tzer (OOPSLA'22)

DetectedBug 49 Confirmed 37 Fixed 25

For best reading experience, you are highly recommended to view [this overview](#) on your web browser.

Get Started

Prerequisites

- OS: A Linux System with Docker Support
- Hardware: X86 CPU; 8GB RAM; 256GB Storage; Good Network to GitHub and Docker Hub;

Before you start, please make sure you have Docker installed.

```
# Test docker availability
sudo docker --version
# Output looks like: (no error)
# Docker version 20.10.12, build #91ed707e
```

Otherwise please follow the [installation page](#) of Docker.

Quick Start with Docker



Image: [tzerbot/oops1a](https://hub.docker.com/r/tzerbot/oops1a)

ganler refactor: bug finding badges 2f217ce 2 minutes ago 80 commits

docs	artifact layout	2 months ago
paper_data	feat: upload original data	3 months ago
src	adopt new tvn	28 days ago
tvn_cov_patch	delete backtick	3 months ago
.gitignore	refact: ignore cache	28 days ago
Dockerfile	refact: repo link modified to ise-uiuc.org	2 months ago
LICENSE	add license	3 months ago
README.md	refact: bug finding badges	2 minutes ago
bug-report.ipynb	refact: repo link modified to ise-uiuc.org	2 months ago
requirements.txt	Upload repo	6 months ago

TZER

DetectedBug 49 Confirmed 37 Fixed 25

arXiv 2202.09947 Open in Colab Image size 1.68 GB Artifact OOPSLA22

Artifact • Reproduce Bugs • Quick Start • Installation • Extend Tzer

Coverage-Guided Tensor Compiler Fuzzing with Joint IR-Pass Mutation

This is the source code repo for "Coverage-Guided Tensor Compiler Fuzzing with Joint IR-Pass Mutation" (accepted by OOPSLA'22).

Releases
2 tags

Packages
No packages published

Contributors 4

- ganler Jiawei Liu
- Tzer-AnonBot
- UniverseFly Yuxiang Wei
- syang-ng Sen Yang

Languages

- Python 84.1%
- Jupyter Notebook 15.4%
- Other 0.5%



GitHub [ise-uiuc/tzer](https://github.com/ise-uiuc/tzer)