Structure Resilience in Greybox Fuzzing via Automated Error Recovery

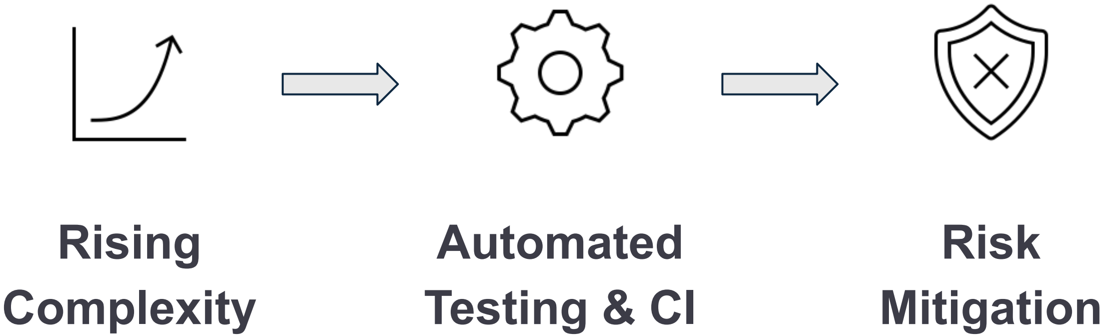
Bachir Bendrissou

Imperial College London

Supervisors: **Cristian Cadar, Alastair Donaldson**

ASE2024 Doctoral Symposium

Critical Role of Software Testing

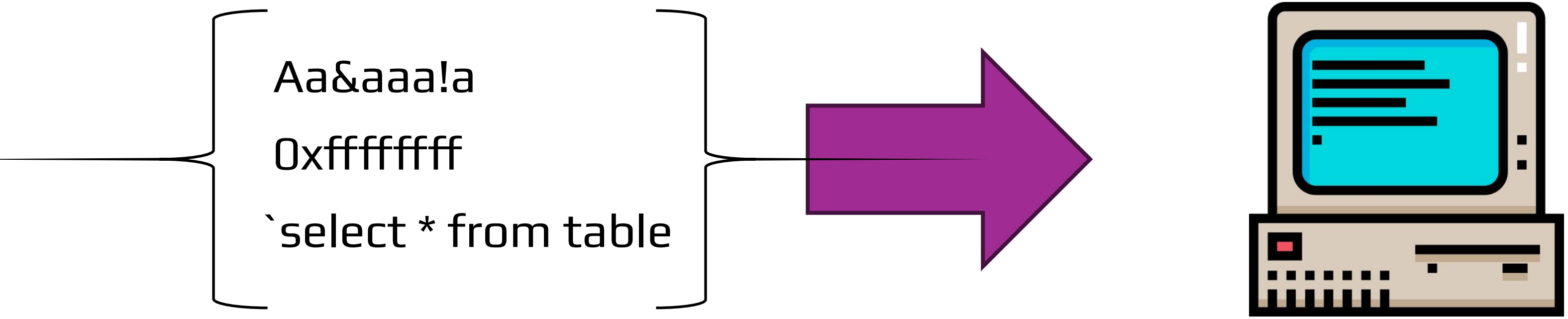


Rising Complexity

Automated Testing & CI

Risk Mitigation

Fuzz Testing (Fuzzing)



Aa&aaa!a

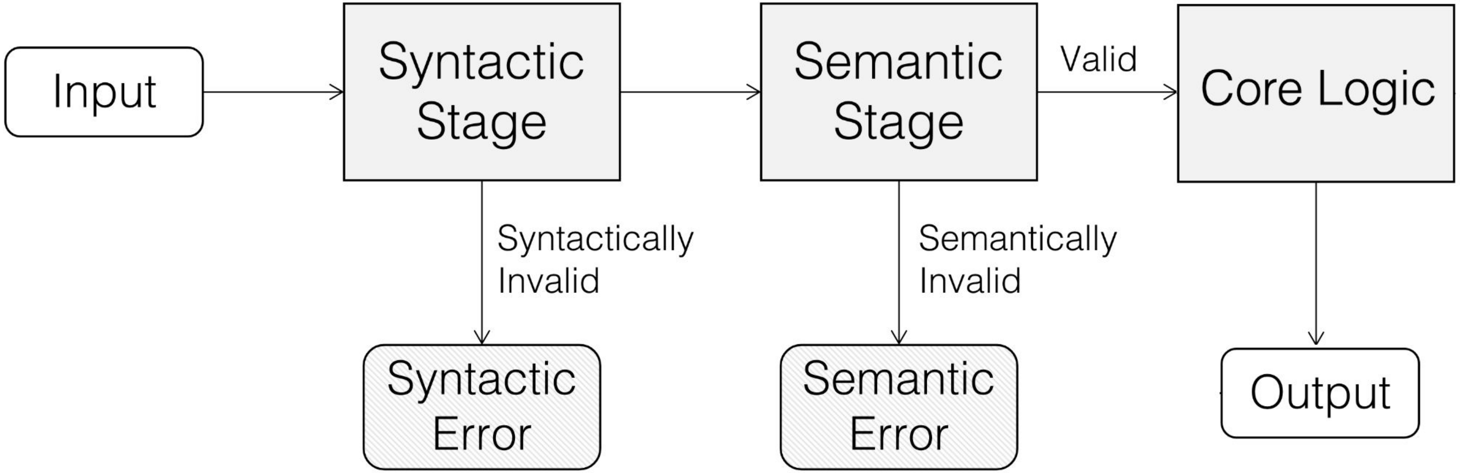
0xffffffff

`select * from table

Main Challenges

1. Test inputs must satisfy input constraints
2. Should exercise a variety of code paths in core program logic

Input Processing Pipeline



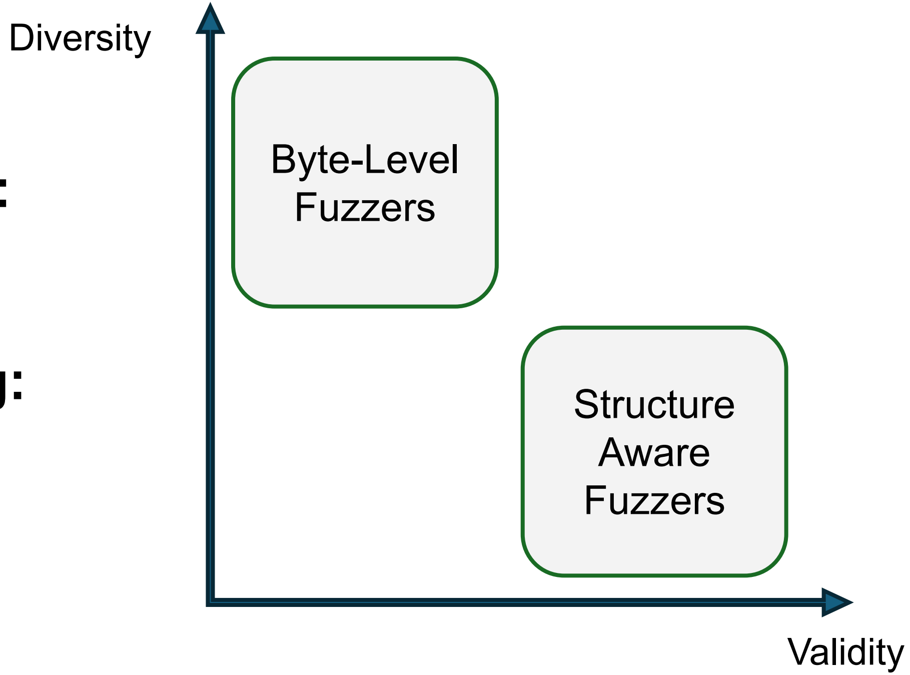
State of the Art

1. **Mutation-based fuzzing:**

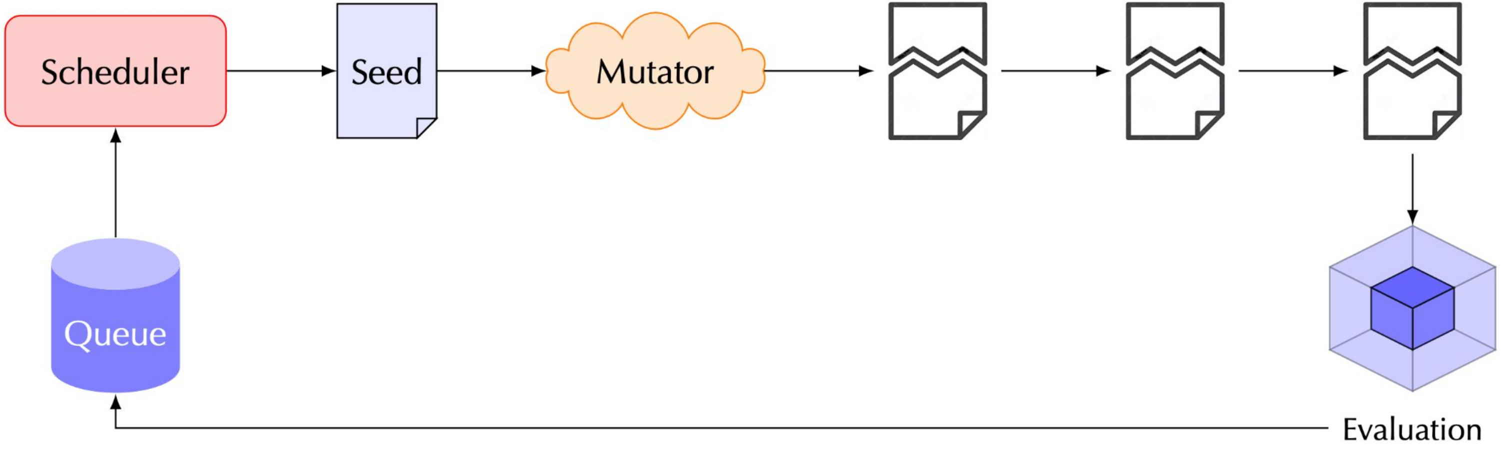
   diverse tests, but invalid

2. **Grammar-based fuzzing:**

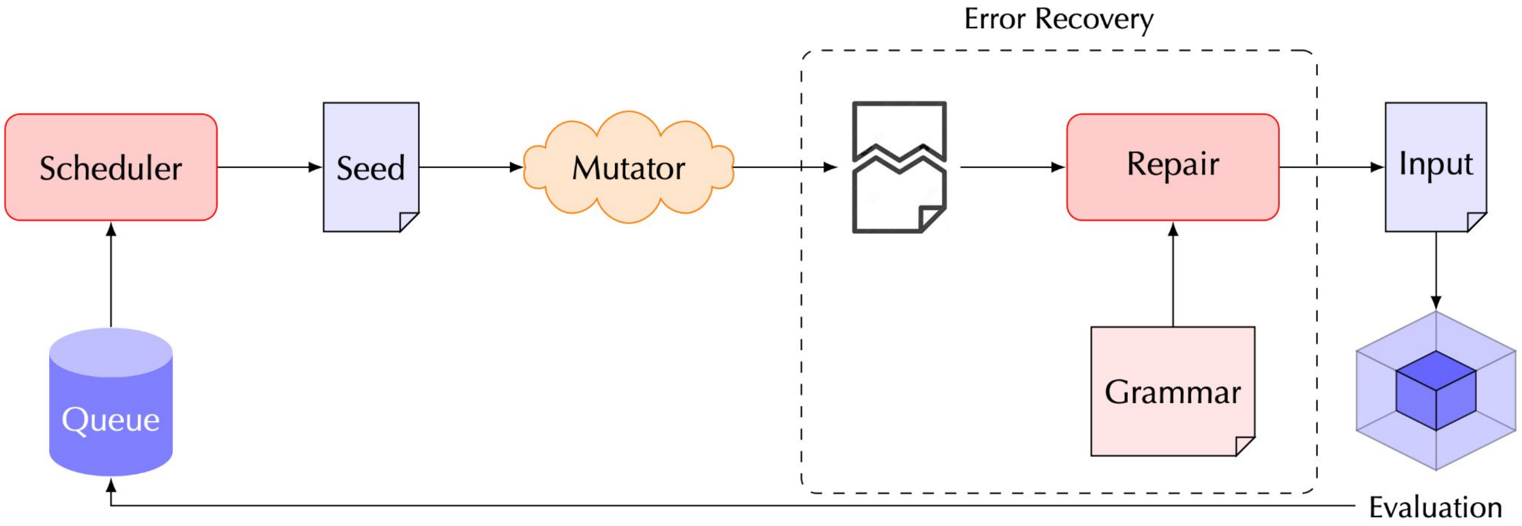
   valid tests, but uniform



State of the Art: American Fuzzy Lop (AFL)



Proposed Solution: AFLRepair



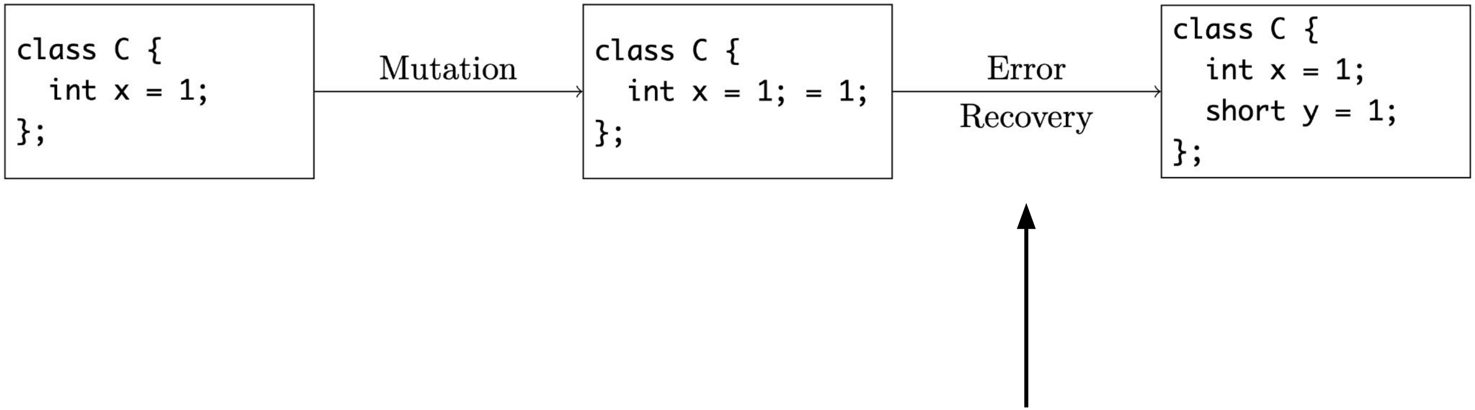
Automatic Error Recovery

1. Originally used to report programming errors and suggest fixes

2. Search-based approach

3. Highly precise and efficient

Automatic Error Recovery



```
class C {
  int x = 1;
};
```

Mutation

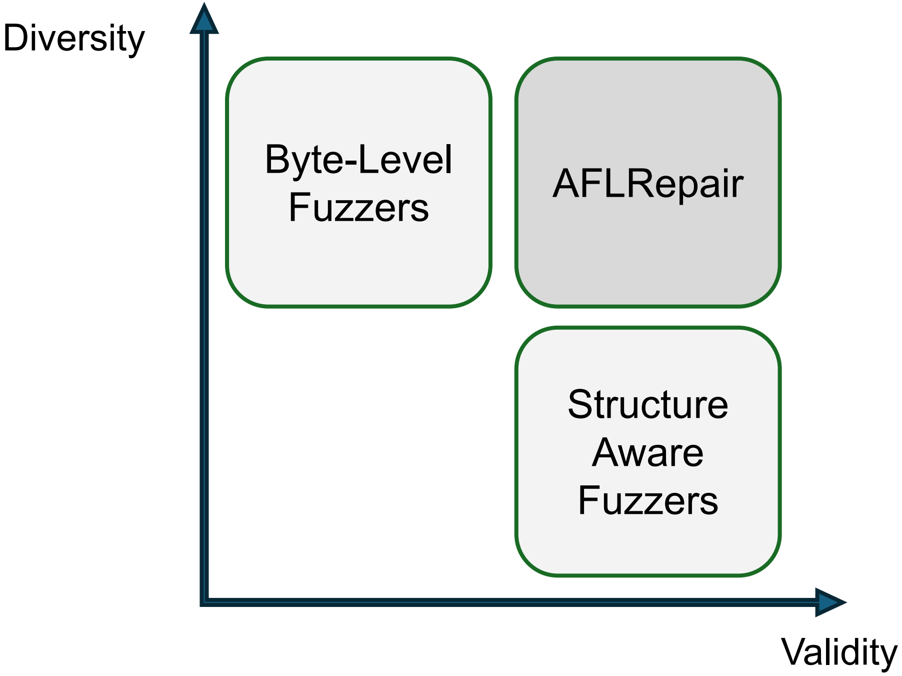
```
class C {
  int x = 1; = 1;
};
```

Error Recovery

```
class C {
  int x = 1;
  short y = 1;
};
```

“Insert Short, Insert <id>”

★ *The Differentiating Factor*



Produce highly diverse test cases while maintaining structure validity

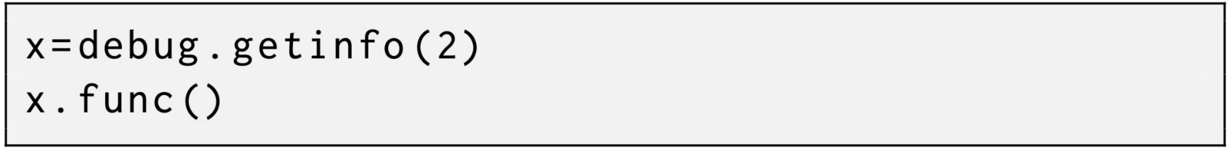
Preliminary Experiment

Setup:

- Systems under test: cJSON, Lua
- Tools: AFL, AFL+GM, AFLRepair

Results:

- Lua crash found by AFLRepair
- Crashing Input:



```
x=debug.getinfo(2)
x.func()
```

Evaluation Plan: Systems Under Test

Program	Language	LOC
Lua	Lua	15 k
LuaJIT	lua	50 k
Ruby	Ruby	768k
V8	JavaScript	1.5M
WAMR	WebAssembly	170K

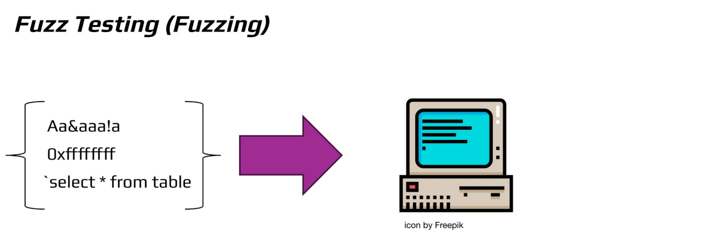
Evaluation Plan: Tools

1. **Nautilus:** Generates and mutates inputs using a grammar
2. **Nautilus+AFLRepair:** Imports Nautilus inputs as seed corpus, and performs AFLRepair mutations

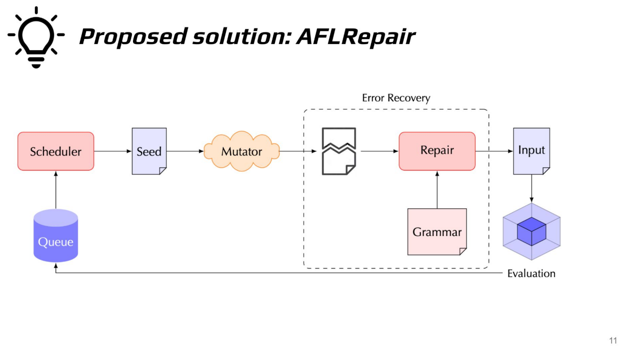
Conclusion

- ❖ Greybox fuzzing is a scalable and effective approach at finding **software vulnerabilities**
- ❖ The challenge is to ensure input **diversity** and **validity**
- ❖ AFLRepair can **achieve both simultaneously**
- ❖ The research capitalizes on current advances in the field

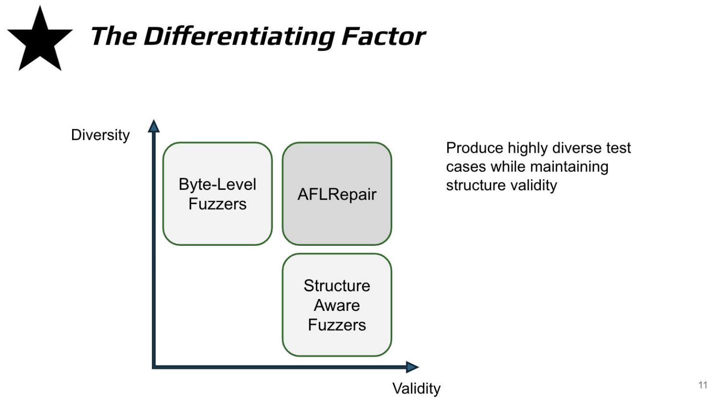
Thank you



Fuzz Testing (Fuzzing)

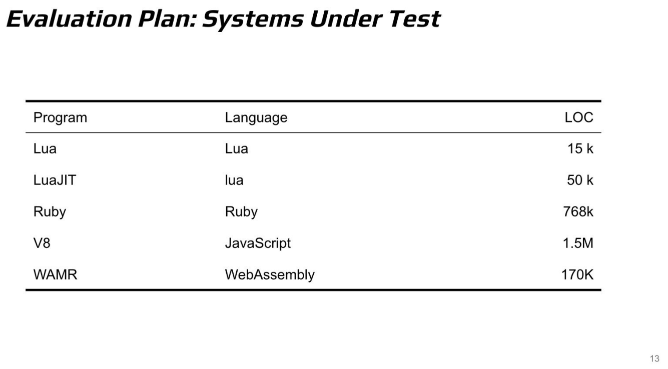


Proposed solution: AFLRepair



The Differentiating Factor

Produce highly diverse test cases while maintaining structure validity



Evaluation Plan: Systems Under Test

Program	Language	LOC
Lua	Lua	15 k
LuaJIT	lua	50 k
Ruby	Ruby	768k
V8	JavaScript	1.5M
WAMR	WebAssembly	170K