Fuzzlang:
Transformer and LLM-Agent for Enhanced Compilation Error Repair

Baodi Shan[1], Barbara Chapman[1], Johannes Doerfert[2]

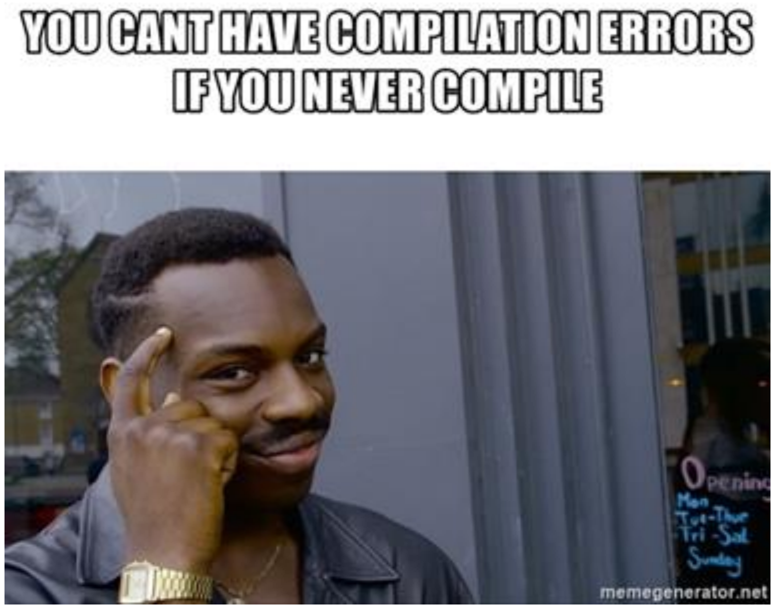
Mar 1, 2025

1 Stony Brook University
2 Lawrence Livermore National Laboratory

Problem Statement

Compilation Error

How to fix compilation errors?



Related Work

Code Fix for C/C++ Compilation Error -- Traditional Methods

- Compiler itself
  - **Detailed Diagnostics**
  - **Enhanced Error Messages**
  - **Fix-it Hints**
- IDE Real Time Feedback

- Static Analysis
- Cross-file Error Contexts
- Incremental Compilation

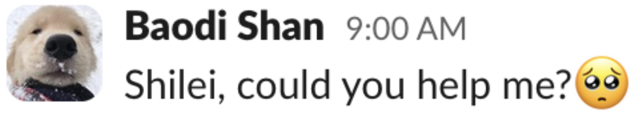
Code Fix for C/C++ Compilation Error -- Traditional Methods

- Compiler itself
  - **Detailed Diagnostics**
  - **Enhanced Error Messages**
  - **Fix-it Hints**
- IDE Real Time Feedback

- Static Analysis
- Cross-file Error Contexts
- Incremental Compilation

Google

stackoverflow



Baodi Shan 9:00 AM

Shilei, could you help me?🥺

Code Fix for C/C++ Compilation Error -- ML Method

- ➔ Deepfix (2017 AAAI)
    - ◆ GRU（Gated Recurrent Unit）
- ➔ TRACER(2018 ICSE-SEET)
    - ◆ RNN
- ➔ DrRepair (2020 ICML), Break-It-Fix-It (2021 ICML)
    - ◆ LSTM
- ➔ ……

https://program-repair.org/tools.html

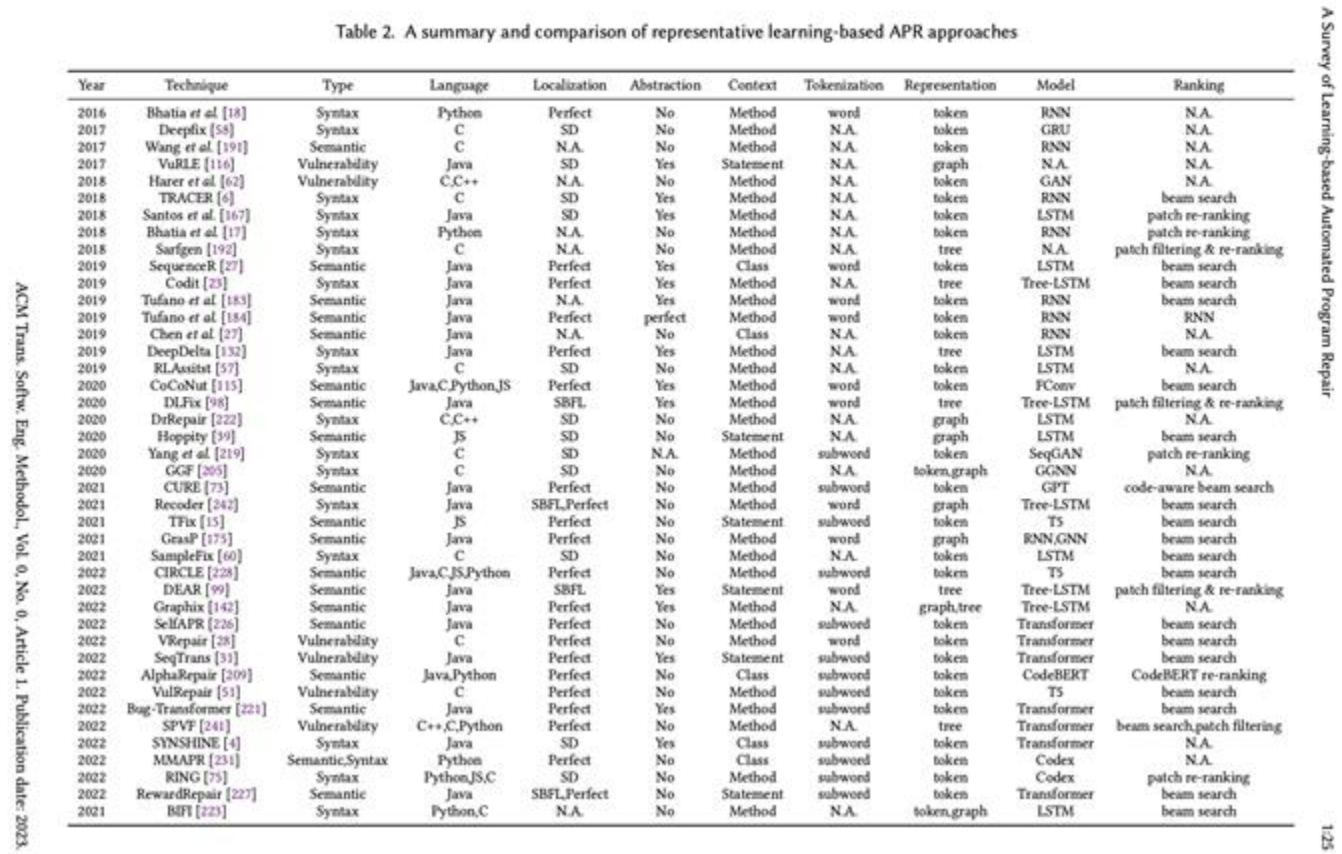


Table 2. A summary and comparison of representative learning-based APR approaches

Year	Technique	Type	Language	Localization	Abstraction	Context	Tokenization	Representation	Model	Ranking
2016	Bhatia *et al.* [18]	Syntax	Python	Perfect	No	Method	word	token	RNN	N.A.
2017	Deepfix [58]	Syntax	C	SD	No	Method	N.A.	token	GRU	N.A.
2017	Wang *et al.* [191]	Semantic	C	N.A.	No	Method	N.A.	token	RNN	N.A.
2017	VuRLE [116]	Vulnerability	Java	SD	Yes	Statement	N.A.	graph	N.A.	N.A.
2018	Harer *et al.* [62]	Vulnerability	C,C++	N.A.	No	Method	N.A.	token	GAN	N.A.
2018	TRACER [6]	Syntax	C	SD	Yes	Method	N.A.	token	RNN	beam search
2018	Santos *et al.* [167]	Syntax	Java	SD	Yes	Method	N.A.	token	LSTM	patch re-ranking
2018	Bhatia *et al.* [17]	Syntax	Python	N.A.	No	Method	N.A.	token	RNN	patch re-ranking
2018	Sarfgen [192]	Syntax	C	N.A.	No	Method	N.A.	tree	N.A.	patch filtering & re-ranking
2019	SequenceR [27]	Semantic	Java	Perfect	Yes	Class	word	token	LSTM	beam search
2019	Codit [23]	Syntax	Java	Perfect	Yes	Method	N.A.	tree	Tree-LSTM	beam search
2019	Tufano *et al.* [183]	Semantic	Java	N.A.	Yes	Method	word	token	RNN	beam search
2019	Tufano *et al.* [184]	Semantic	Java	Perfect	perfect	Method	word	token	RNN	RNN
2019	Chen *et al.* [27]	Semantic	Java	N.A.	No	Class	N.A.	token	RNN	N.A.
2019	DeepDelta [132]	Syntax	Java	Perfect	Yes	Method	N.A.	tree	LSTM	beam search
2019	RLAssitst [57]	Syntax	C	SD	No	Method	N.A.	token	LSTM	N.A.
2020	CoCoNut [115]	Semantic	Java,C,Python,JS	Perfect	Yes	Method	word	token	FConv	beam search
2020	DLFix [98]	Semantic	Java	SBFL	Yes	Method	word	tree	Tree-LSTM	patch filtering & re-ranking
2020	DrRepair [222]	Syntax	C,C++	SD	No	Method	N.A.	graph	LSTM	N.A.
2020	Hoppity [39]	Semantic	JS	SD	No	Statement	N.A.	graph	LSTM	beam search
2020	Yang *et al.* [219]	Syntax	C	SD	N.A.	Method	subword	token	SeqGAN	patch re-ranking
2020	GGF [205]	Syntax	C	SD	No	Method	N.A.	token,graph	GGNN	N.A.
2021	CURE [73]	Semantic	Java	Perfect	No	Method	subword	token	GPT	code-aware beam search
2021	Recoder [242]	Syntax	Java	SBFL,Perfect	No	Method	word	graph	Tree-LSTM	beam search
2021	TFix [15]	Semantic	JS	Perfect	No	Statement	subword	token	T5	beam search
2021	GrasP [175]	Semantic	Java	Perfect	No	Method	word	graph	RNN,GNN	beam search
2021	SampleFix [60]	Syntax	C	SD	No	Method	N.A.	token	LSTM	beam search
2022	CIRCLE [228]	Semantic	Java,C,JS,Python	Perfect	No	Method	subword	token	T5	beam search
2022	DEAR [99]	Semantic	Java	SBFL	Yes	Statement	word	tree	Tree-LSTM	patch filtering & re-ranking
2022	Graphix [142]	Semantic	Java	Perfect	Yes	Method	N.A.	graph,tree	Tree-LSTM	N.A.
2022	SelfAPR [226]	Semantic	Java	Perfect	No	Method	subword	token	Transformer	beam search
2022	VRepair [28]	Vulnerability	C	Perfect	No	Method	word	token	Transformer	beam search
2022	SeqTrans [31]	Vulnerability	Java	Perfect	Yes	Statement	subword	token	Transformer	beam search
2022	AlphaRepair [209]	Semantic	Java,Python	Perfect	No	Class	subword	token	CodeBERT	CodeBERT re-ranking
2022	VulRepair [51]	Vulnerability	C	Perfect	No	Method	subword	token	T5	beam search
2022	Bug-Transformer [221]	Semantic	Java	Perfect	Yes	Method	subword	token	Transformer	beam search
2022	SPVF [241]	Vulnerability	C++,C,Python	Perfect	No	Method	N.A.	tree	Transformer	beam search,patch filtering
2022	SYNSHINE [4]	Syntax	Java	SD	Yes	Class	subword	token	Transformer	N.A.
2022	MMAPR [231]	Semantic,Syntax	Python	Perfect	No	Class	subword	token	Codex	N.A.
2022	RING [75]	Syntax	Python,JS,C	SD	No	Method	subword	token	Codex	patch re-ranking
2022	RewardRepair [227]	Semantic	Java	SBFL,Perfect	No	Statement	subword	token	Transformer	beam search
2021	BIFI [223]	Syntax	Python,C	N.A.	No	Method	N.A.	token,graph	LSTM	beam search

Quanjun Zhang, Chunrong Fang, Yuxiang Ma, Weisong Sun, and Zhenyu Chen. 2023. A Survey of Learning-based Automated Program Repair. ACM Trans. Softw. Eng. Methodol. 33, 2, Article 55 (February 2024), 69 pages. https://doi.org/10.1145/3631974

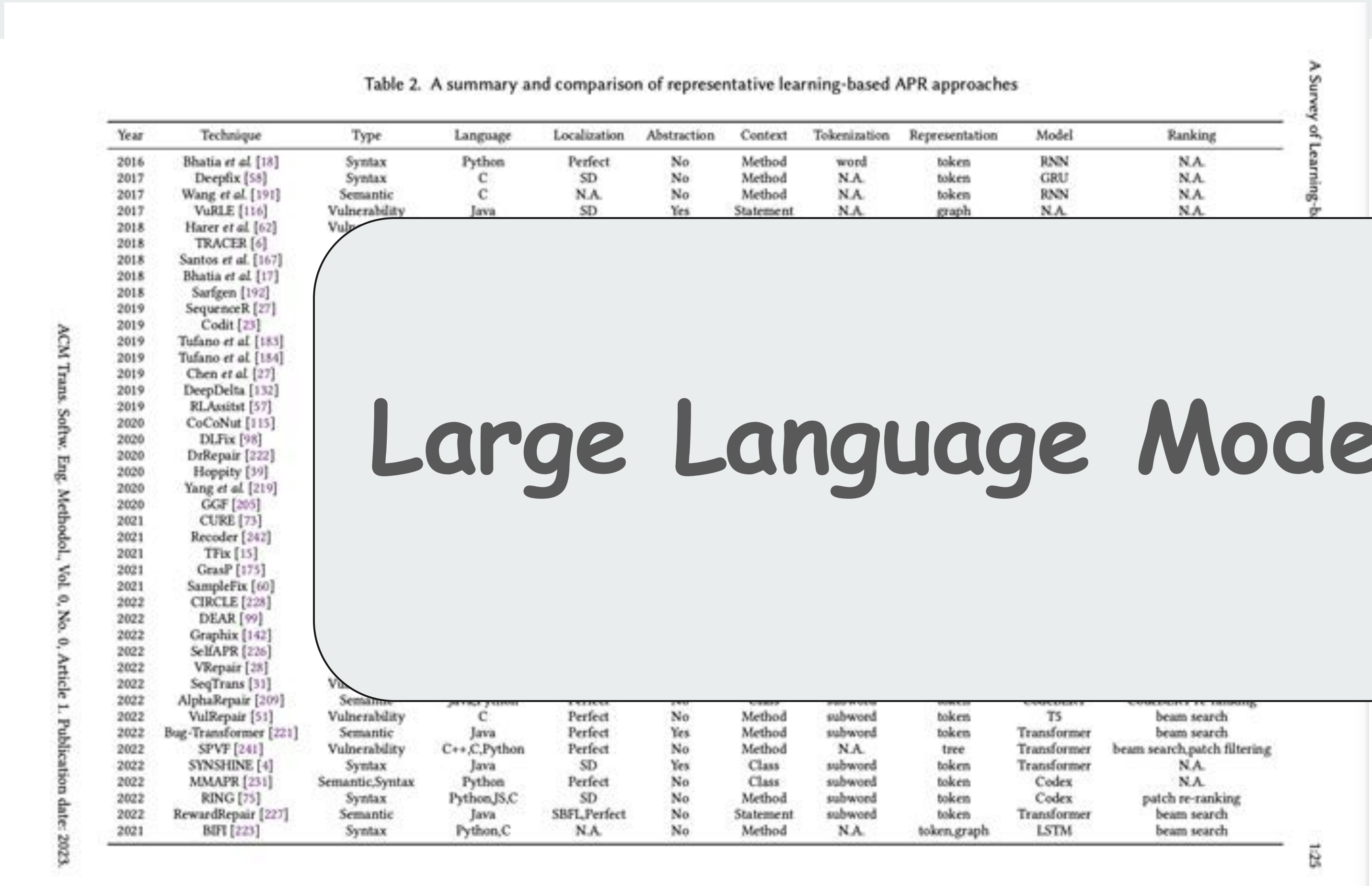


Table 2. A summary and comparison of representative learning-based APR approaches

Year	Technique	Type	Language	Localization	Abstraction	Context	Tokenization	Representation	Model	Ranking
2016	Bhatia *et al.* [18]	Syntax	Python	Perfect	No	Method	word	token	RNN	N.A.
2017	Deepfix [58]	Syntax	C	SD	No	Method	N.A.	token	GRU	N.A.
2017	Wang *et al.* [191]	Semantic	C	N.A.	No	Method	N.A.	token	RNN	N.A.
2017	VuRLE [116]	Vulnerability	Java	SD	Yes	Statement	N.A.	graph	N.A.	N.A.
2018	Harer *et al.* [62]	Vul								
2018	TRACER [6]									
2018	Santos *et al.* [167]									
2018	Bhatia *et al.* [17]									
2018	Sarfgen [192]									
2019	SequenceR [27]									
2019	Codit [23]									
2019	Tufano *et al.* [183]									
2019	Tufano *et al.* [184]									
2019	Chen *et al.* [27]									
2019	DeepDelta [132]									
2019	RLAssitst [57]									
2020	CoCoNut [115]									
2020	DLFix [98]									
2020	DrRepair [222]									
2020	Hoppity [39]									
2020	Yang *et al.* [219]									
2020	GGF [205]									
2021	CURE [73]									
2021	Recoder [242]									
2021	TFix [15]									
2021	GrasP [175]									
2021	SampleFix [60]									
2022	CIRCLE [228]									
2022	DEAR [99]									
2022	Graphix [142]									
2022	SelfAPR [226]									
2022	VRepair [28]									
2022	SeqTrans [31]	Vu								
2022	AlphaRepair [209]	Semantic								
2022	VulRepair [51]	Vulnerability	C	Perfect	No	Method	subword	token	T5	beam search
2022	Bug-Transformer [221]	Semantic	Java	Perfect	Yes	Method	subword	token	Transformer	beam search
2022	SPVF [241]	Vulnerability	C++,C,Python	Perfect	No	Method	N.A.	tree	Transformer	beam search,patch filtering
2022	SYNSHINE [4]	Syntax	Java	SD	Yes	Class	subword	token	Transformer	N.A.
2022	MMAPR [231]	Semantic,Syntax	Python	Perfect	No	Class	subword	token	Codex	N.A.
2022	RING [75]	Syntax	Python,JS,C	SD	No	Method	subword	token	Codex	patch re-ranking
2022	RewardRepair [227]	Semantic	Java	SBFL,Perfect	No	Statement	subword	token	Transformer	beam search
2021	BIFI [223]	Syntax	Python,C	N.A.	No	Method	N.A.	token,graph	LSTM	beam search

Large Language Model

Quanjun Zhang, Chunrong Fang, Yuxiang Ma, Weisong Sun, and Zhenyu Chen. 2023. A Survey of Learning-based Automated Program Repair. ACM Trans. Softw. Eng. Methodol. 33, 2, Article 55 (February 2024), 69 pages. https://doi.org/10.1145/3631974

For LLMs, we need data

- ➔ Deepfix -- Homework
- ➔ C-Pack-IPAs -- Homework
- ➔ CodeForces Dataset -- Online Judge(OJ)

- Stack Overflow? Github? Other Public Resources?

In clang, how many compilation error types?

3541 unique error types! (commit 6441df3b)

C++/OpenMP/OpenACC/OpenCL are developing, more error types are coming!

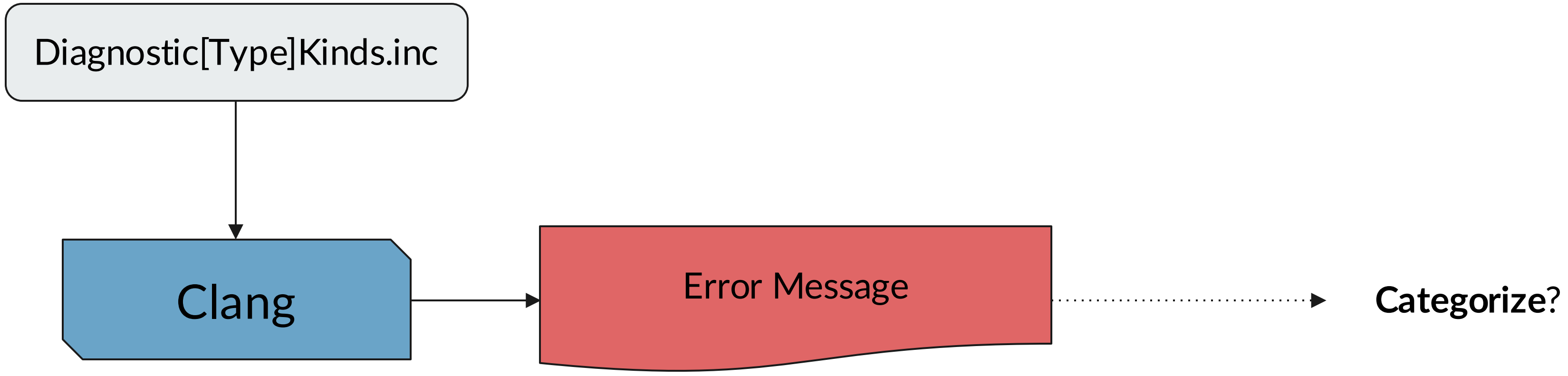
So, aims of Fuzzlang

- Let LLMs understand compilation error
- Categorize compilation error types
- Generate more compilation error

- ❑ Reproduce compilation error
- ❑ Cover more error types
- ❑ Fine-tuning current LLM

Implementation - Foundational Components

How clang diagnostic engine works?



Error Message?

expected %0 - “**err_expected**”

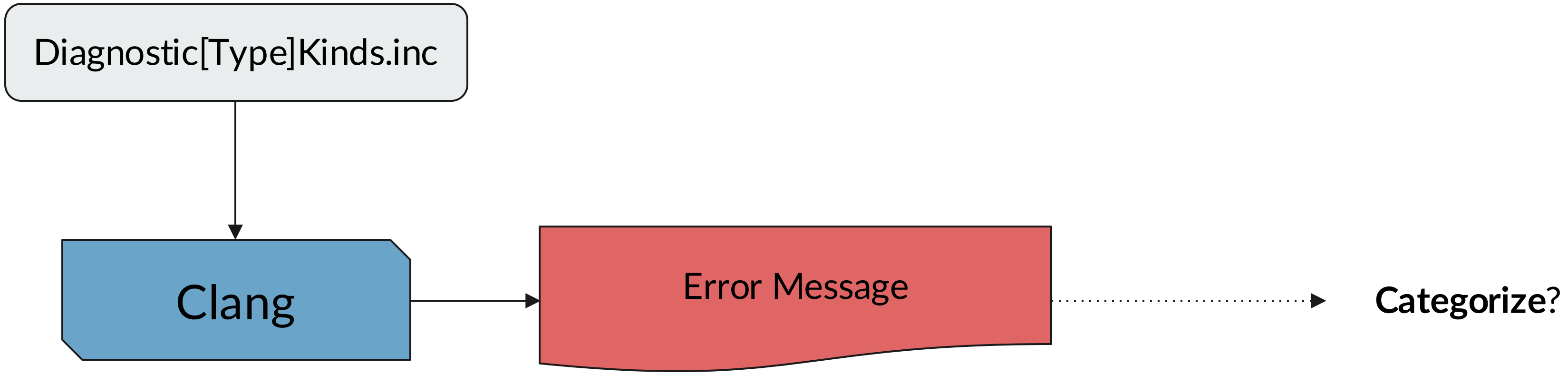
expected %0 or %1 - “**err_expected_either**”

expected %1 after %0 - “**err_expected_after**”

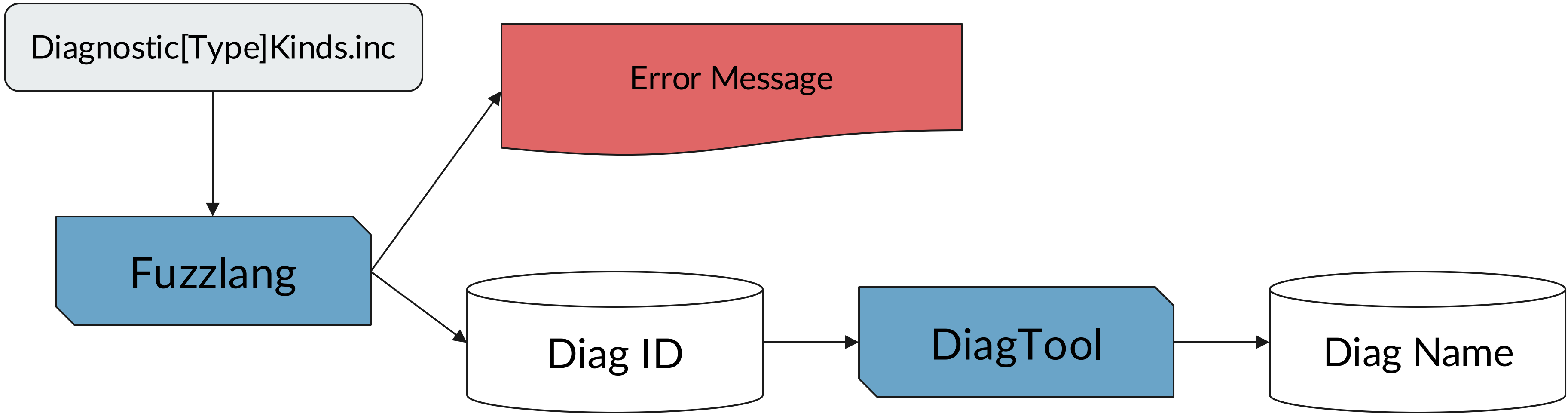
Fuzzlang Foundational Components

- Specially Modified *Clang* -- Diag ID
- Specially Modified *Diagtool* -- Diag Name

How clang diagnostic engine works?

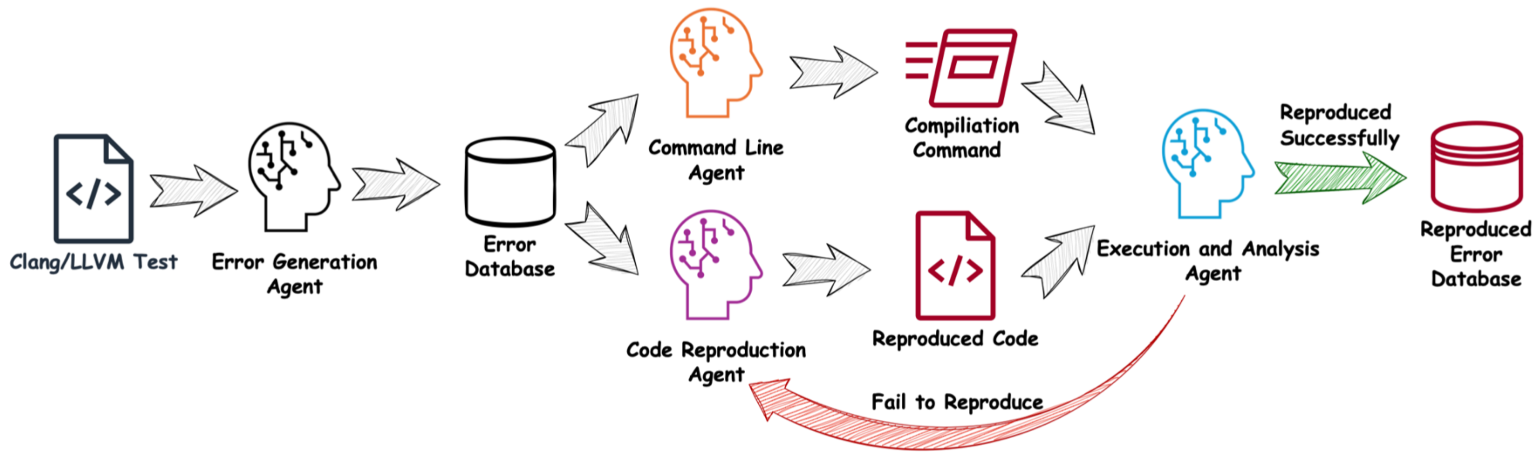


How Fuzzlang diagnostic engine works?



Implementation - Fuzzlang Agent

Workflow of Fuzzlang Agent



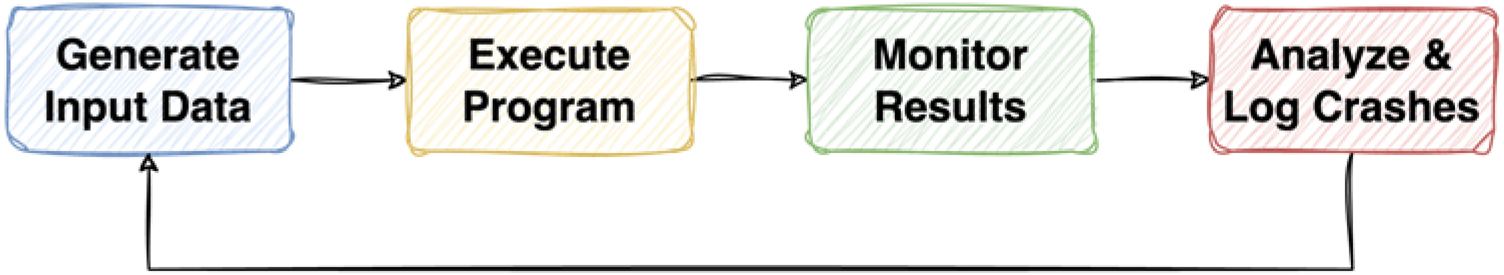
Implementation - Fuzzlang Transformer

Introduction of Fuzzing

What's Fuzzing?

Automated software testing technique that inputs invalid, unexpected, or random data into a program to find vulnerabilities and errors.

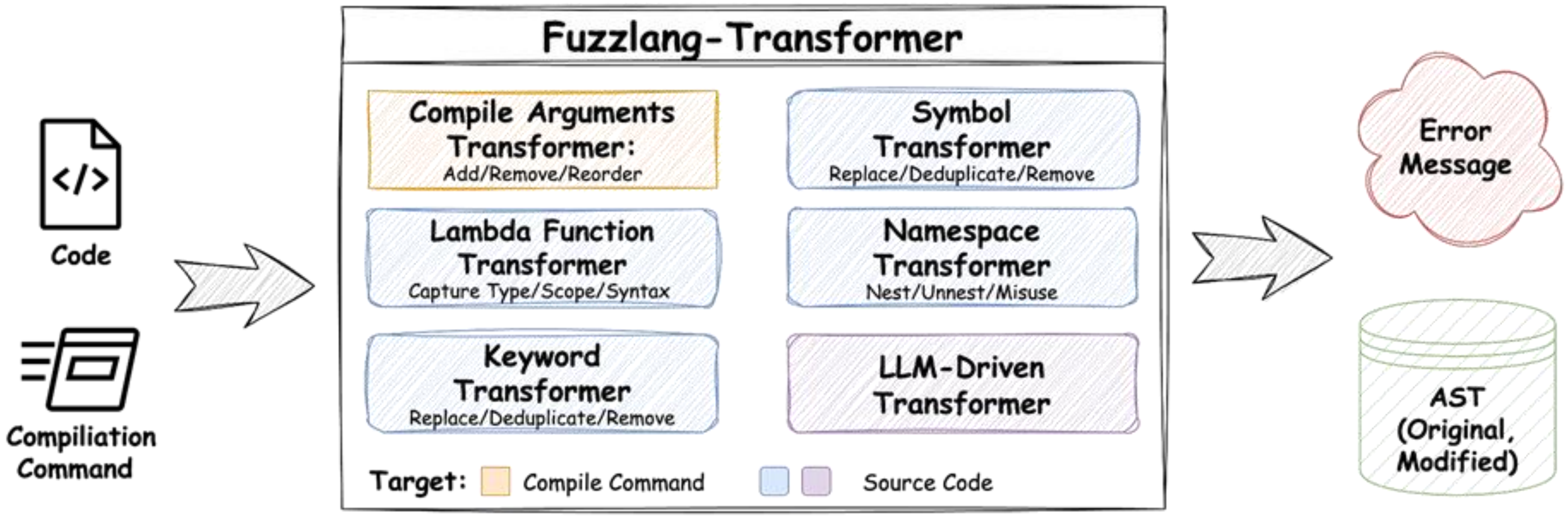
Workflow of Fuzzing?



Design of Fuzzlang Transformer

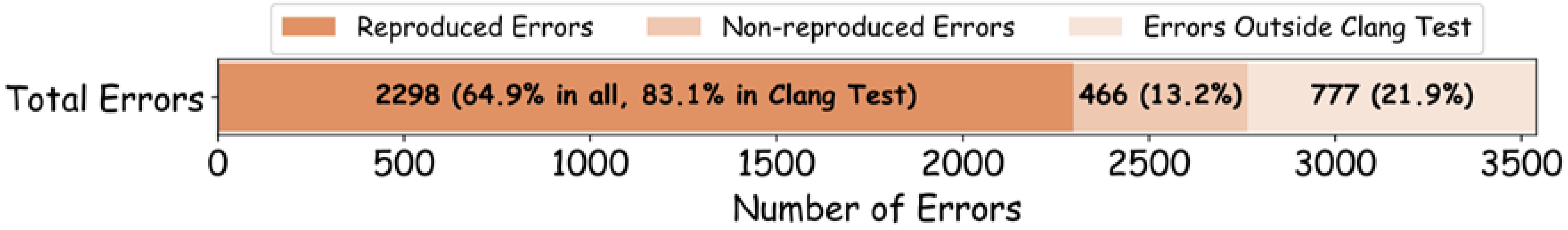
- A Clang Wrapper (Used as a C/C++ compiler)
- “Modification” Modules
    - Compilation Command
    - Source Code
- Collecting Error Message & AST

Workflow of Fuzzlang Transformer

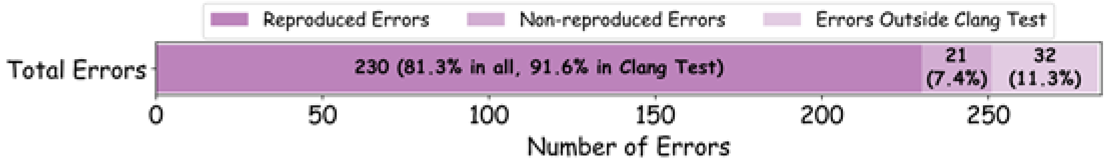


Evaluation

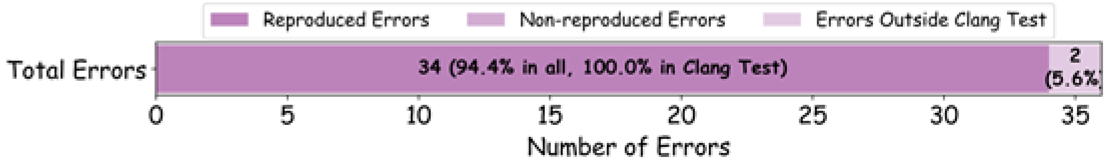
Error Diagnostics Reproducing Breakdown



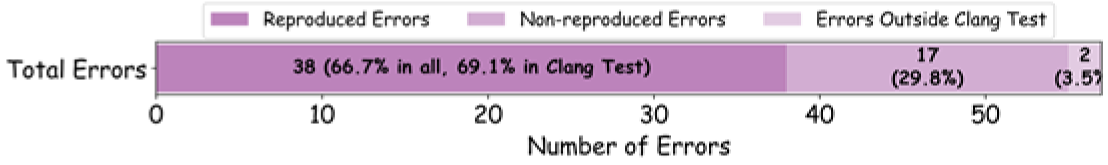
Bonus: Fuzzlang for HPC



(a) OpenMP Error Diagnostics Breakdown



(b) OpenACC Error Diagnostics Breakdown

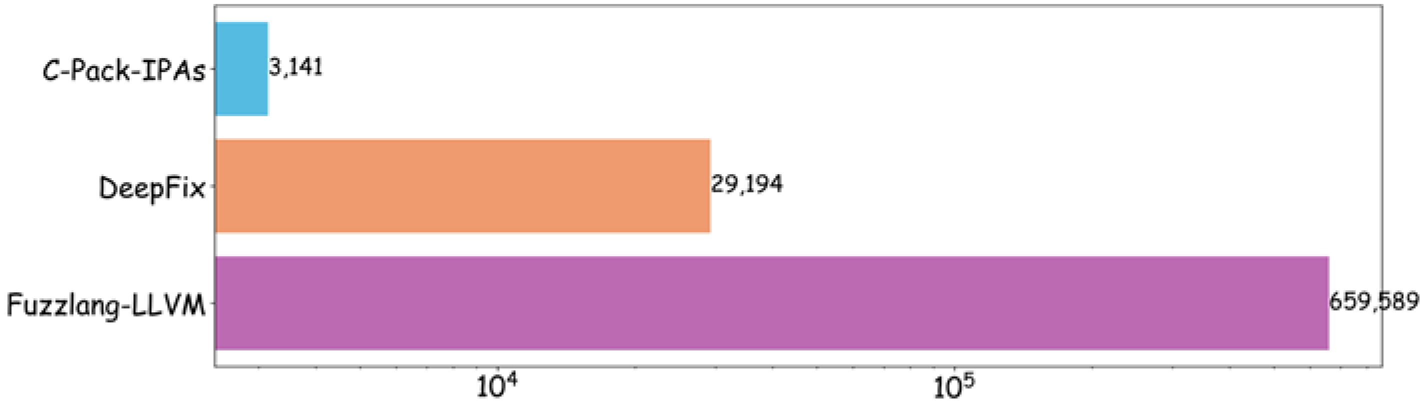


(c) OpenCL Error Diagnostics Breakdown

Dataset: Fuzzlang-LLVM

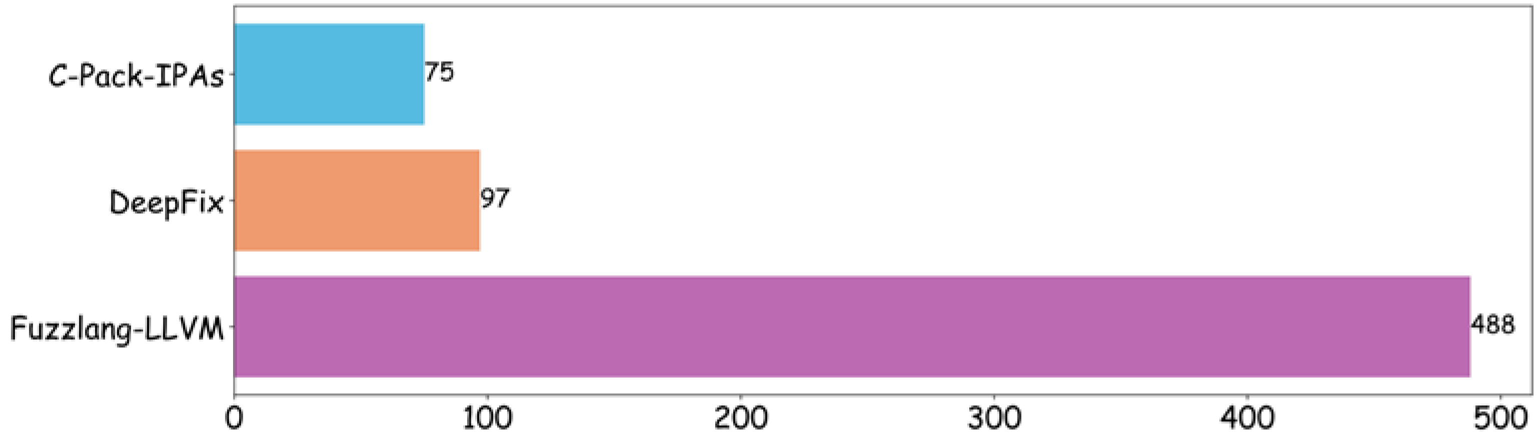
- Compile LLVM project(LLVM+Clang)
- ~4000 Compilation Steps
- 659k errors
- 488 Unique Types

Dataset: Fuzzlang-LLVM



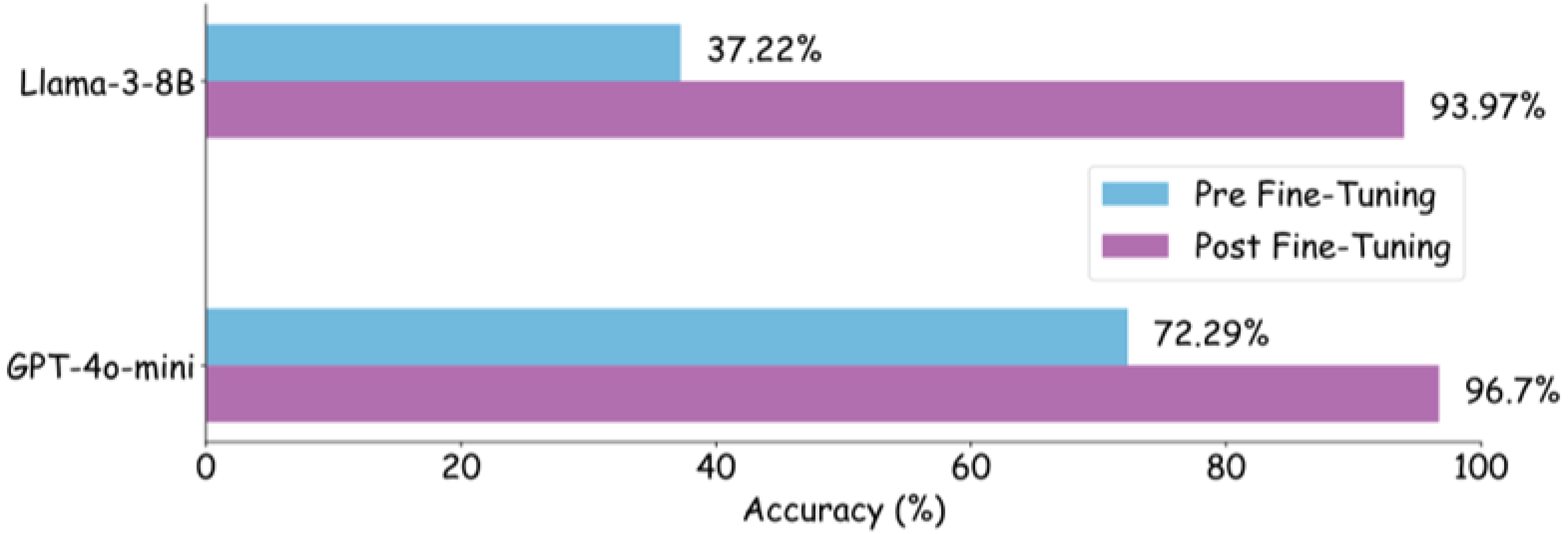
(a) Total Number of Errors (log)

Dataset: Fuzzlang-LLVM



(b) Number of Independent Error Types

Validation -- Fine Tuning LLM



Next Steps

What next?

→ Larger dataset from other C/C++ project

→ Modification cross files

- Runtime Error Monitoring(Vulnerability)

Q&A

Contact: baodi.shan@stonybrook.edu