Detecting Bugs in Rust Compiler Fix Suggestions via Constraint-Violation-Guided Mutation

FSE 2026

Zixi Liu, Yang Feng, Jialiang Jiang, Baowen Xu — Nanjing University

Rust Programming Language

- Memory safety without a garbage collector: ownership, borrowing and lifetimes
- Enforced statically — the compiler rejects code it cannot prove “memory safe”
  - Technically, memory safe in the sense that it adheres to Rust’s rules for memory safety
- Rust’s strict rules are often difficult to grasp
  - The compiler provides fix suggestions that are supposed to guide developers towards correct Rust code

Rust Compiler (rustc) Fix Suggestions

```rust
fn add(a: i32, b: i32) -> i32 {
    a + b
}


fn main(){
    let x: u32 = 5;
    let y: u32 = 10;
    add(x, y);
}
```

```
error[E0308]: arguments to this function are incorrect
 --> tmp.rs:9:5
  |
9 |     add(x, y);
  |     ^^^ -  - expected `i32`, found `u32`
  |         |
  |         expected `i32`, found `u32`
  |
note: function defined here
 --> tmp.rs:1:4
  |
1 | fn add(a: i32, b: i32) -> i32 {
  |    ^^^ ------  ------
help: you can convert a `u32` to an `i32` and panic if the converted value doesn't fit
  |
9 |     add(x.try_into().unwrap(), y);
  |         ++++++++++++++++++++
help: you can convert a `u32` to an `i32` and panic if the converted value doesn't fit
  |
9 |     add(x, y.try_into().unwrap());
  |            ++++++++++++++++++++

error: aborting due to 1 previous error
```

Incorrect Fix Suggestions (Compiler Bug)

```
1 struct S<'a>(&'a ());
2 fn f(s: S<'_>) -> _ { s }
3
```

```
1 error[E0121]: the placeholder `_` is not allowed
2 within types on item signatures for return types
3 help: replace with the correct return type:
4         `S<'static>`
```

Incorrect Suggestion

‘static is global lifetime.
The suggestion would result in incorrect fix that will not compile!

The correct fix

```
1 struct S<'a>(&'a ());
2 fn f(s: S<'_>) -> S<'_> { s }
```

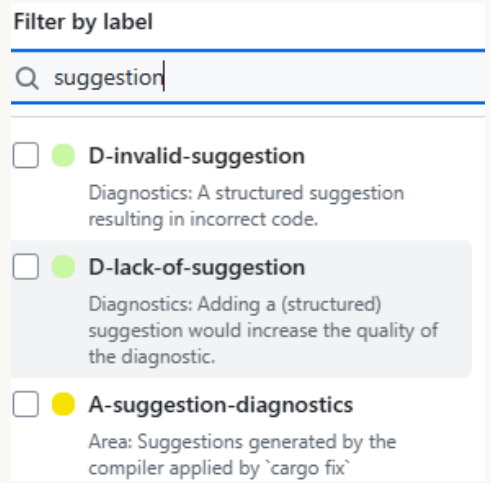
Key Motivation: Incorrect Fix Suggestions

- Diagnostic issues account for roughly 20% of all reported compiler bugs
- More than 95% of those were discovered by the Rust team or by ordinary users — not by any automated testing tool
- Existing rustc fuzzers optimise for crashes and miscompilation
- Current rustc tooling are inadequate in detecting/fixing these issues

Cross Module Reasoning

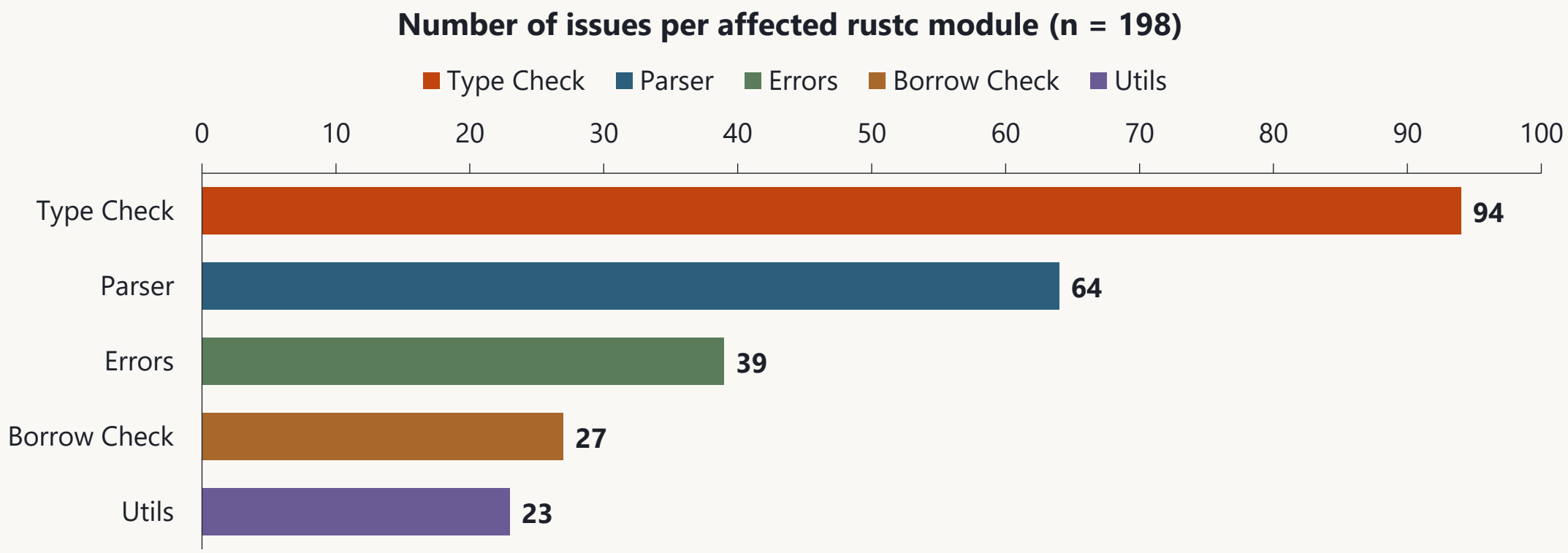
- 198 closed rustc issues, Sept 2022 – Sept 2025,
  - labelled D-invalid-suggestion, A-suggestion-diagnostics or D-lack-of-suggestion.

- ~25% (46/193) of fixes involve more than one module (cross-module)



Modules touched by one fix	1	2	3	4	5	6	7	Total
Number of issues	**152**	26	12	3	2	2	1	**198**

Types of Suggestion Bugs



Existing Tools (Baseline)

Tool	How it generates programs	What it looks for
Fuzz-rustc	Mutates the input byte stream (LibFuzzer)	Crashes
Tree-splicer	Recombines AST fragments from seeds	Crashes
ICEMaker	Both, plus Miri and Clippy	Crashes, miscompilation
RustSmith	Generates valid programs from the grammar	Miscompilation (differential)
Rustlantis	Emits custom MIR via mir!()	MIR mis-optimisation

Some other existing tools were excluded due to infeasibility in comparison (tied to specific rustc version, unavailable source code)

(e.g., Rust-twin, Typecheck-fuzzer)

SugBreaker

Take valid Rust programs. Deliberately break the rules Rust cares about.

Then check whether rustc's own suggested fix actually fixes the program.

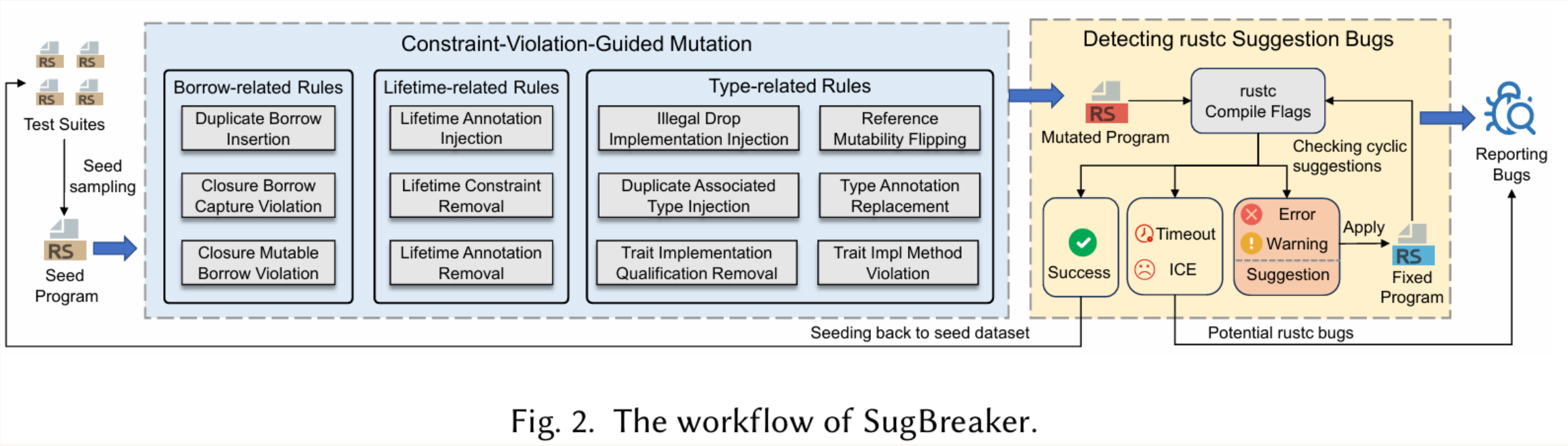
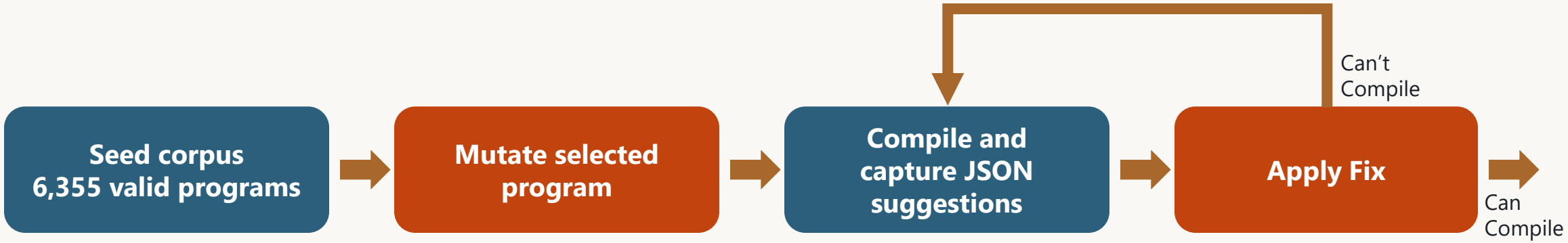


Fig. 2. The workflow of SugBreaker.

General Process of SugBreaker



Programs that compile successfully are returned to the seed corpus, so the corpus grows as fuzzing proceeds.
The apply-and-recompile step repeats for up to m = 6 rounds (empirically chosen).

Compiles

The mutant, or the fixed program, is added back to the seed corpus

Crashes or hangs

An ICE, a segfault, or a compile taking over 10 seconds is reported directly as a bug.

Never compiles

If the error survives six rounds of applying rustc's own advice, the suggestion is suspect.

Mutate rustc's test suite

- Select "compilable" test suite from rustc project (Some tests are purposefully not compilable)
- Mutate valid compilable programs to incur fix suggestions
- Apply Fix suggestions to see if compiler has suggested a "correct" fix

- Three class of mutations
  - Borrow-Related Rust Program Mutation (Total: 3)
  - Lifetime-Related Rust Program Mutation (Total: 3)
  - Type-Related Rust Program Mutation (Total: 6)

  * We will only cover one mutation per each class

Borrow Mutation: Duplicate Borrow Insertion

Before Mutation

```
1 let mut value = 10;
2 let borrow = &mut value;
3 println!("{}", borrow);
```

After Mutation

```
1 let mut value = 10;
2 let borrow = &mut value;
3 let borrow_dup = &mut value;
4 println!("{}", borrow);
```

What is wrong

Rust allows either **one** mutable reference or **multiple** immutable references
The following mutation creates a duplicate mutable reference

Lifetime Mutation: Lifetime Annotation Injection

Before Mutation

```
1 struct myStruct<'a> {a: &'a i32}
2 impl <'a> myStruct <'a>{
3   fn method(&self, x: &'a i32) {}
4 }
```

After Mutation

```
1 struct myStruct<'a> {a: &'a i32}
2 impl <'a> myStruct <'a>{
3   fn method<'a>(&self, x: &'a i32) {}
4 }
```

What is wrong

Adding a redundant lifetime annotation <'a> can create inconsistencies or ambiguities in the lifetime

Type Mutation: Illegal Drop Implementation Injection

Before Mutation

```
1 trait MyTrait {}
2 struct MyStruct {}
```

After Mutation

```
1 trait MyTrait {}
2 impl Drop for MyTrait{
3     fn drop(&mut self) {}
4 }
5 struct MyStruct {}
6 impl Drop for &'_ mut MyStruct{
7     fn drop(&mut self) {}
8 }
9
```

What is wrong

Drop runs a destructor when a value goes out of scope, so it needs a concrete type that owns its data. A trait is not a type, and a reference does not own what it points to — neither has anything to destroy.

Rust Compiler (rustc) Fix Suggestions

- Rust provides an automatic fixing tool cargo-fix
- Suggestions carries an applicability level: MachineApplicable, MaybeIncorrect, HasPlaceholders, Unspecified
- cargo fix automatically applies **only** the MachineApplicable ones


```
{
  "message": "you can convert a `u32` to an `i32` and panic if the converted value doesn't fit",
  "code": null,
  "level": "help",
  "spans": [
    {
      "file_name": "tmp.rs",
      "byte_start": 107,
      "byte_end": 107,
      "line_start": 9,
      "line_end": 9,
      "column_start": 10,
      "column_end": 10,
      "is_primary": true,
      "text": [
        {
          "text": "    add(x, y);",
          "highlight_start": 10,
          "highlight_end": 10
        }
      ],
      "label": null,
      "suggested_replacement": ".try_into().unwrap()",
      "suggestion_applicability": "MachineApplicable",
      "expansion": null
    }
  ],
  "children": [],
  "rendered": null
}
```


JSON formatted error messaging by rustc (--error-format=json)

Automatically Applying Fix Suggestions

- Cannot use cargo-fix as it doesn't work on all fix suggestions
- SugBreaker parses the JSON diagnostic output and applies the fix
- Crucially, all types of fixes are applied

Notably, the paper does not clearly mention the correctness of the parsing and fix applying mechanism

Fix Suggestion Bugs

Cyclic

rustc contradicts itself: it applies a fix, then undoes it in the next iteration.

Unidentifiable location

The fix points outside user-level source code — into a macro expansion or rustc's own libraries.

Ineffective

The same error survives every round. The fix never addressed it.

Suspicious

Fix results in a different error. Automation cannot judge it, so it is flagged for manual inspection.

Did it partially fix, introduce new error, or is incorrect?

Real Bugs Identified in rustc 1.87.0

5
Fixed by the Rust team

6
Confirmed

1
Reported, still open

What kind of bugs

9 are genuine fix-suggestion bugs
2 are internal compiler errors (ICEs)
1 is a non-terminating compile, classified by the Rust team as a Hang

Where they live

Fixes involve HIR analysis; MIR borrow checker and HIR-based type checking.

Demonstrates SugBreaker's coverage in triggering bugs in various Rust core modules

Performance of SugBreaker (vs. baseline tools)

Line coverage of rustc	Fuzz-rustc	Tree-splicer	ICEMaker	Rustlantis	RustSmith	SugBreaker
rustc_ast (parser)	71.1	**72.7**	45.2	30.5	27.7	64.8
rustc_trait (trait solving)	52.2	66.9	43.8	16.7	16.2	**67.7**
rustc_borrowck (borrow check)	33.9	50.5	28.2	20.0	26.3	**61.2**
rustc_errors (diagnostics)	59.7	62.7	58.9	39.9	40.2	**67.0**
Total	55.4	**58.2**	45.6	29.6	28.6	57.5

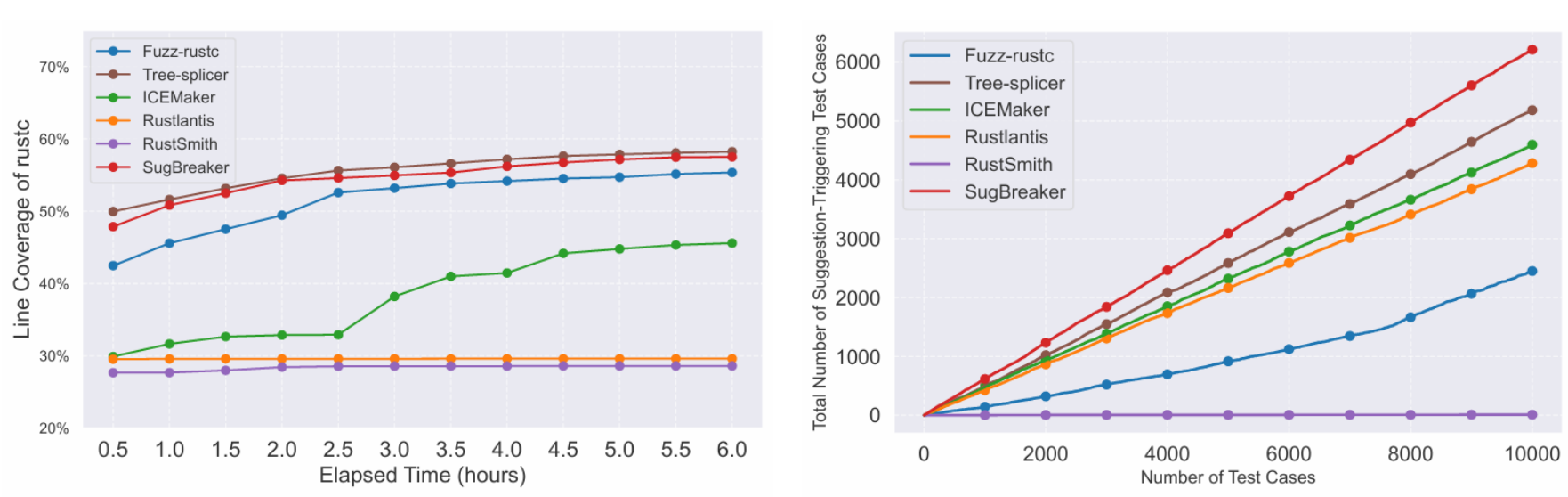
Tree-splicer edges the total (58.2% vs 57.5%) by splicing random AST fragments — parser coverage, not checker coverage.

The authors argue HIR- and MIR-level coverage is what finds suggestion bugs — but never demonstrate that link.

Ablation Study

Module	Seed	Lifetime		Borrow		Type	
		cov	Δ cov	cov	Δ cov	cov	Δ cov
rustc_ast_passes	59.59	61.40	+1.81	59.59	–	59.59	–
rustc_hir_analysis	59.16	59.34	+0.18	59.47	+0.31	68.24	+9.08
rustc_hir_typeck	52.75	52.75	–	58.18	+5.43	65.65	+12.90
rustc_infer/src/infer	81.85	82.14	+0.29	83.94	+2.09	85.38	+3.53
rustc_middle/src/traits	51.58	51.58	–	58.30	+6.72	64.82	+13.24
rustc_middle/src/ty	69.91	70.04	+0.13	72.59	+2.68	76.31	+6.40
rustc_mir_build/src/builder	80.88	80.88	–	84.12	+3.24	84.16	+3.28
rustc_borrowck	65.82	65.96	+0.14	71.69	+5.87	71.31	+5.49
rustc_errors	58.88	60.72	+1.84	65.10	+6.22	65.90	+7.02
rustc_hir_analysis/src/errors	0.00	0.00	–	0.00	–	44.62	+44.62
rustc_trait_selection/src/error_reporting	0.00	0.00	–	80.95	+80.95	80.95	+80.95
rustc_borrowck/src/diagnostics	1.00	1.00	–	29.58	+28.58	34.11	+33.11

Efficiency of SugBreaker



(a) The trend of `rustc` line coverage over time.

(b) The suggestion trigger rate over test case.

Fig. 3. Efficiency comparison between SugBreaker and the baseline approaches.

Actual bugs Identified

```
#[derive(Clone, Debug)]
struct Point<T> {  v: T  }

fn main() {
    let a: Point<u8> = dbg!(Point { v: 42 });
    let b: Point<u8> = dbg!(&a);
}
```

```
error[E0308]: mismatched types
  |
6 |  let b: Point<u8> = dbg!(&a);
  |                     ^^^^^^^ expected `Point<u8>`, found `&Point<u8>`
help: consider using clone here
 --> /home/xxx/.rustup/toolchains/…/…/library/std/src/macros.rs:367:20
    |
367 |             tmp.clone()
    |                 ++++++++

   6 |    let b: Point<u8> = dbg!(a.clone());
     |                               +++++++
```

(a) A fix suggestion's bug triggered by error E0308.

```
pub struct Point<A> { x: A }
pub trait MyTrait {
    fn fun(&self, p: Point<A>);
}
fn main(){}
```

```
error[E0412]: cannot find type `A` in this scope
  |
3 |     fn fun(&self, p: Point<A>);
  |                           ^ not found in this scope
help: there is an enum variant `brotli::enc::threading::
InternalSendAlloc::A` and 4 others; try using the variant's enum
  |
3 -     fn fun(&self, p: Point<A>);
3 +     fn fun(&self, p: Point<futures_0_1_31::future::Either>);

2 | pub trait MyTrait<A> {
  |                  ++
```

(b) A fix suggestion's bug triggered by error E0412.

```
trait Trait {}
struct W<T>(T);

impl<T, U> Trait for W<()> where
W<T>: Trait, W<U>: Trait {}

fn impls<T: Trait>() {}

fn main() { impls::<W<_>>(); }
```

```
error[E0207]: the type parameter `T` is not constrained by the
impl trait, self type, or predicates
 --> main.rs:4:6
  |
4 | impl<T, U> Trait for W<()> where W<T>: Trait, W<U>: Trait {}
  |      ^ unconstrained type parameter
  |
4 | impl<T, U> Trait for W<()> where W<T>: Trait, W<U>: Trait {}
  |         ^ unconstrained type parameter
```

(c) A timeout issue triggered by error E0207.

Error Incorrectly points to rust standard library

Suggests spurious enum instead of adding <A> to the Trait

Timeout due to infinite loop during trait resolution

Personal Thoughts

Verifying Automated Fixing

The paper never asks whether SugBreaker itself applies fixes incorrectly. A failed recompile is blamed on rustc by default.

False Positives

SugBreaker found real compiler bugs. But how many cases did it flag for inspection, and how many of those were real?

Taxonomy of Mutation

No case study names the mutation that produced it. We cannot tell whether SugBreaker found these samples, or which rule did.