KU LEUVEN

DistriNet

Migrating from unsafe languages to Rust

Alicia Andries

alicia.andries@kuleuven.be

Migrating to Rust

Linux

Android

Facebook

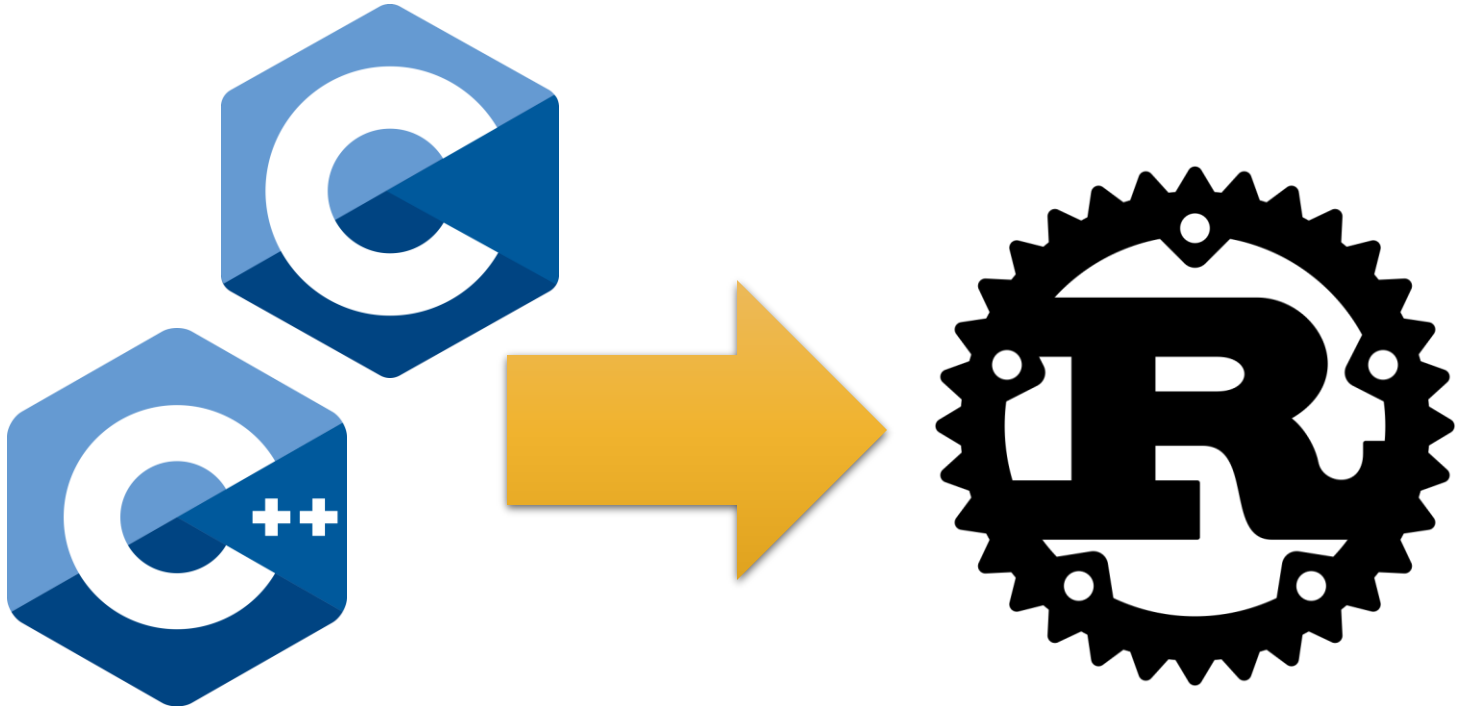
Amazon

Microsoft

Mozilla

Coursera

...



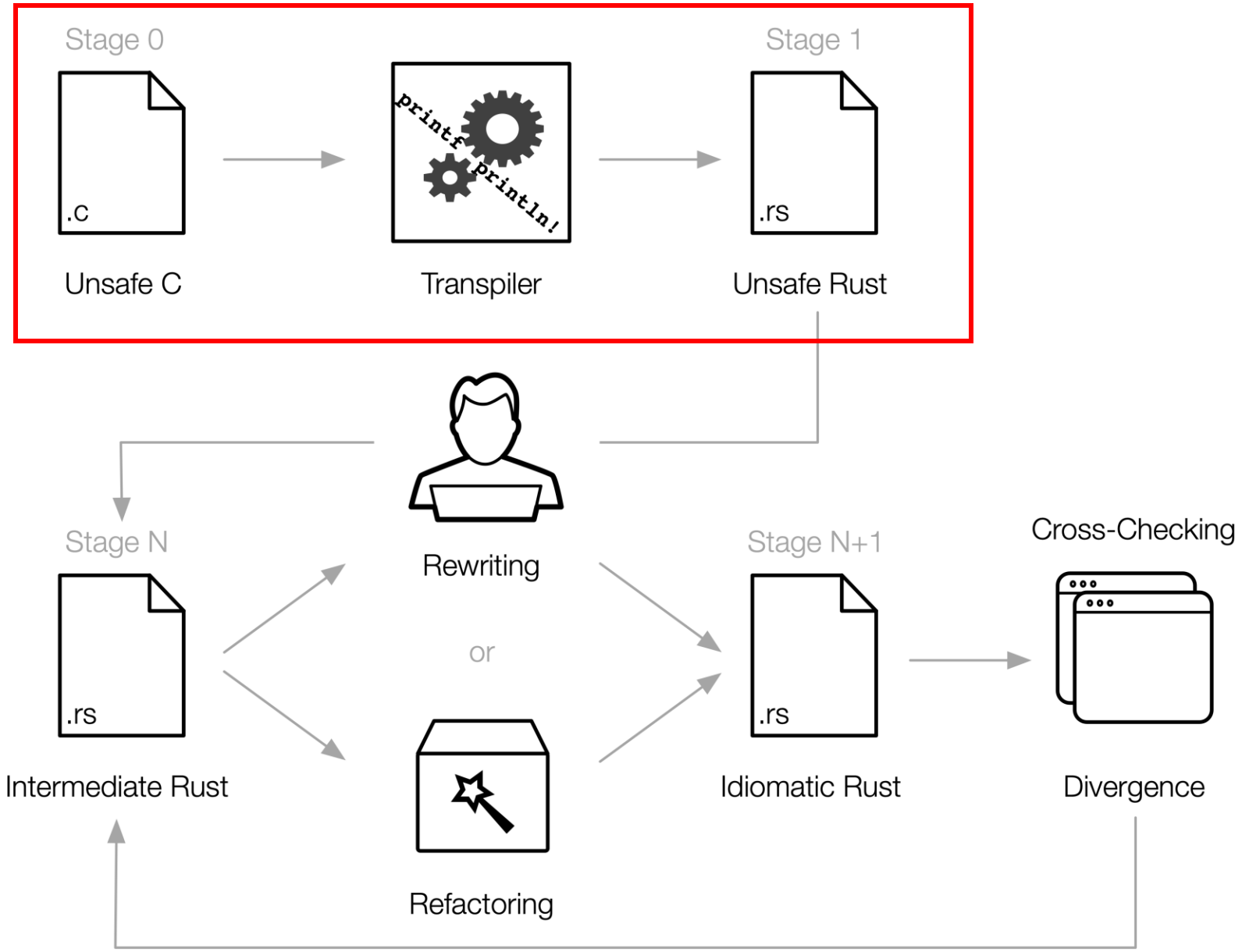

Legacy code

Automatic Translation

- Crust
- Corrode

C2Rust

immunant


Stage 0
.c
Unsafe C
printf println!
Transpiler
Stage 1
.rs
Unsafe Rust
Rewriting
Stage N
.rs
Intermediate Rust
or
Refactoring
Stage N+1
.rs
Idiomatic Rust
Cross-Checking
Divergence

C

```c
void insertion_sort(int const n, int *
const p) {

  for (int i = 1; i < n; i++) {
    int const tmp = p[i];
    int j = i;
    while (j > 0 && p[j-1] > tmp) {
      p[j] = p[j-1];
      j--;
    }
    p[j] = tmp;
  }
}
```

Non-idiomatic Rust

```rust
pub unsafe extern "C" fn insertion_sort(n: libc::c_int, p:
*mut libc::c_int) {
  let mut i: libc::c_int = 1 as libc::c_int;

  while i < n {
    let tmp: libc::c_int = *p.offset(i as isize);
    let mut j: libc::c_int = i;
    while j > 0 as libc::c_int
          && *p.offset((j - 1 as libc::c_int) as isize) > tmp
      {
        *p.offset(j as isize) = *p.offset((j - 1 as
            libc::c_int) as isize);
        j -= 1;
      }
    *p.offset(j as isize) = tmp;
    i += 1;
  }
}
```

Remove *Unsafe*

Idiomatic Rust

```
fn insertion_sort(p: &mut Vec<i32>) {
  let mut i = 1;
  while i < p.len() {
    let tmp = p[i];
    let mut j = i;
    while j > 0 && p[j - 1] > tmp {
      p[j] = p[j - 1];
      j -= 1;
    }
    p[j] = tmp;
    i += 1;
  }
}
```

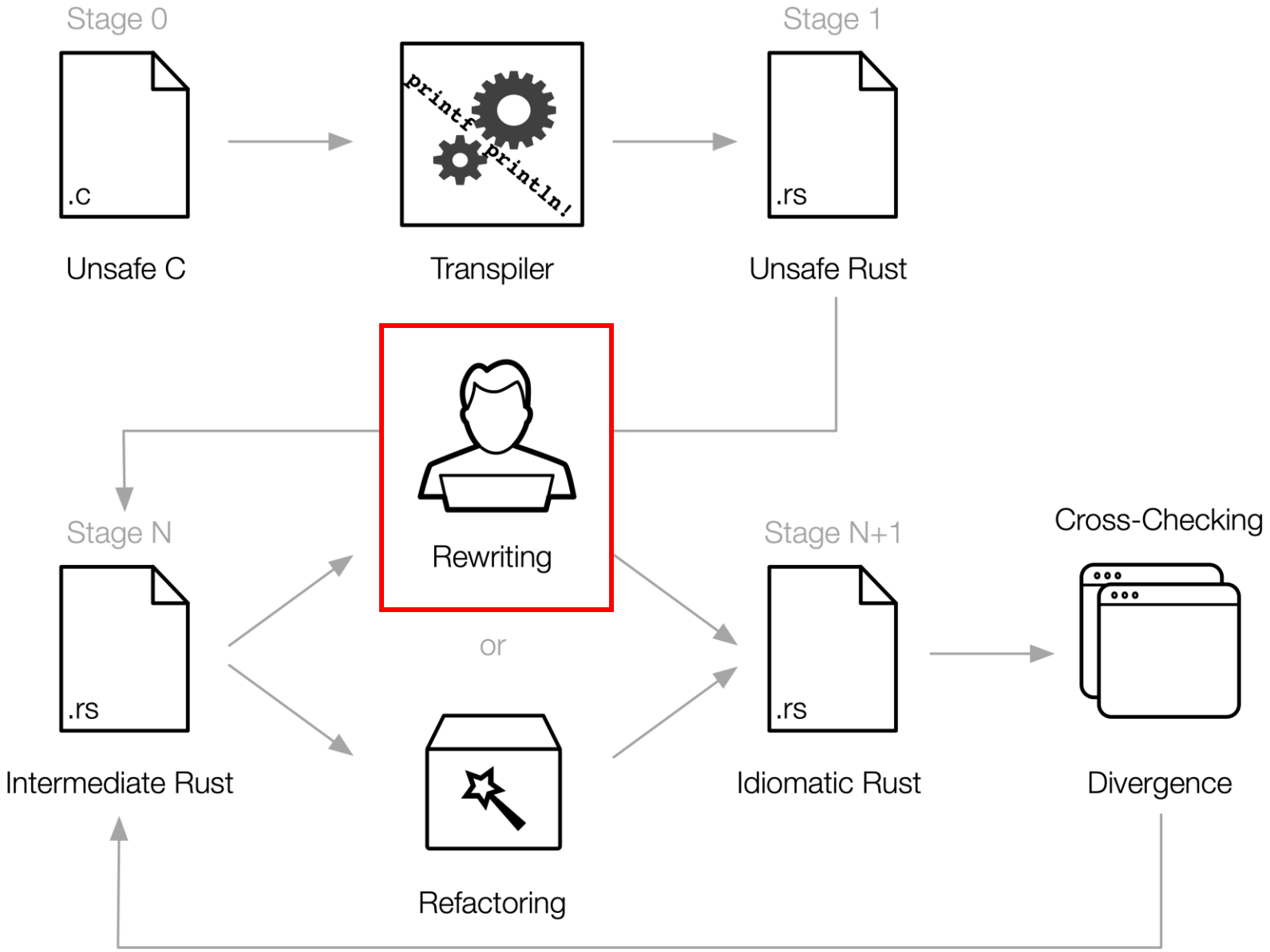
Non-idiomatic Rust

```
pub unsafe extern "C" fn insertion_sort(n: libc::c_int, p:
*mut libc::c_int) {
  let mut i: libc::c_int = 1 as libc::c_int;

  while i < n {
    let tmp: libc::c_int = *p.offset(i as isize);
    let mut j: libc::c_int = i;
    while j > 0 as libc::c_int
          && *p.offset((j - 1 as libc::c_int) as isize) > tmp
      {
        *p.offset(j as isize) = *p.offset((j - 1 as
            libc::c_int) as isize);
        j -= 1;
      }
    *p.offset(j as isize) = tmp;
    i += 1;
  }
}
```

Idiomatic Rust

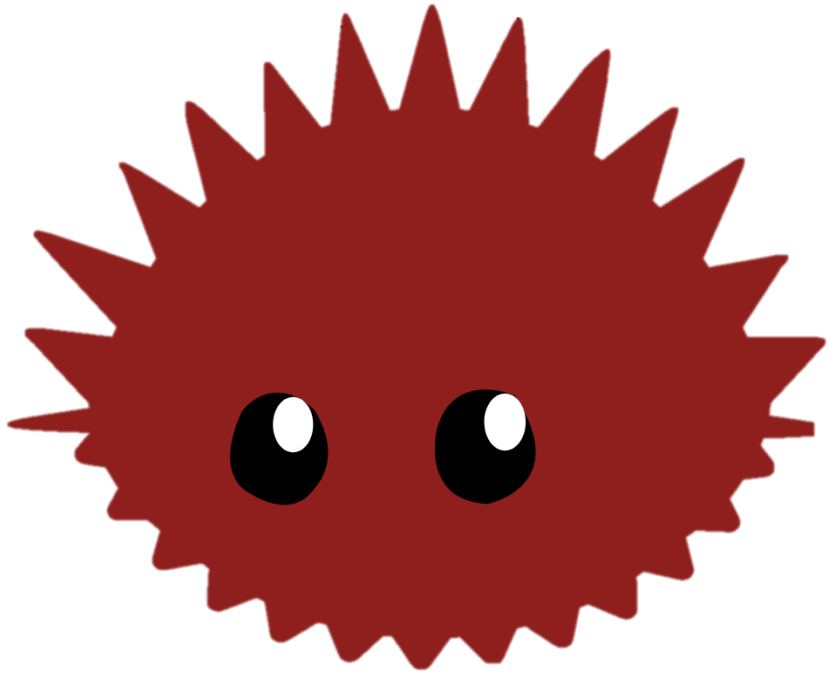
```
fn insertion_sort(p: &mut Vec<i32>) {
  let mut i = 1;
  while i < p.len() {
    let tmp = p[i];
    let mut j = i;
    while j > 0 && p[j - 1] > tmp {
      p[j] = p[j - 1];
      j -= 1;
    }
    p[j] = tmp;
    i += 1;
  }
}
```

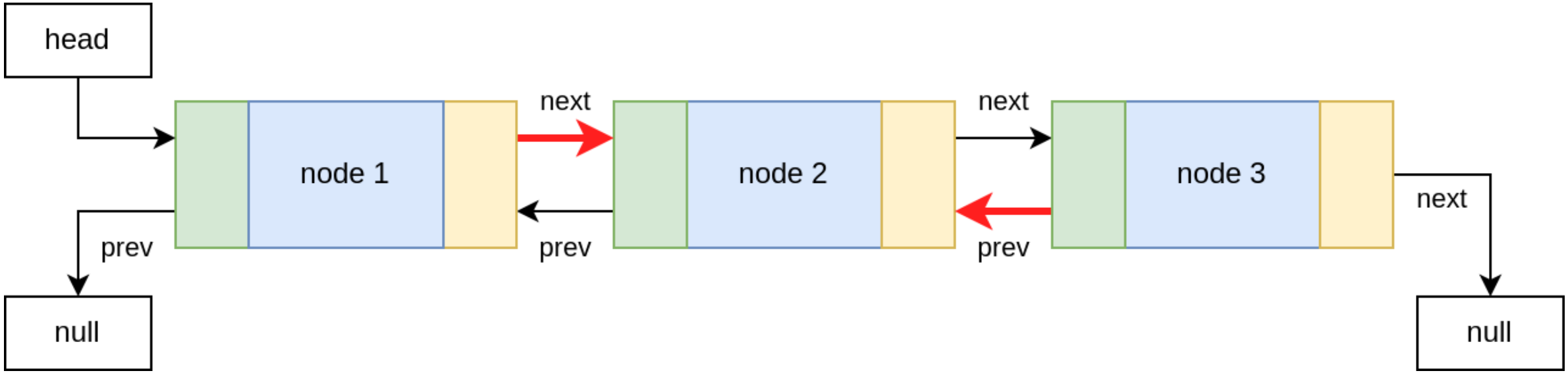

Stage 0
.c
Unsafe C
printf println!
Transpiler
Stage 1
.rs
Unsafe Rust
Rewriting
Stage N
.rs
Intermediate Rust
or
Refactoring
Stage N+1
.rs
Idiomatic Rust
Cross-Checking
Divergence

C2Rust

Semantically identical

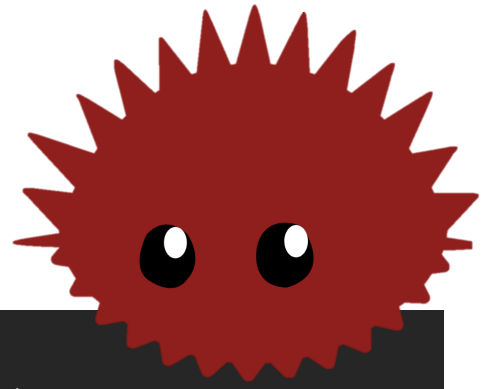
- Unsafe
- Non-idiomatic





https://rust-unofficial.github.io/too-many-lists/

Unsafe Superpowers



1. Dereferencing a raw pointer
2. Calling an unsafe function or method
3. Accessing or modifying a mutable static variable
4. Implementing an unsafe trait
5. Accessing fields of `unions`

```
fn main() {
  let x= Box::<i32>::(4);
  let raw_pointer = Box::into_raw(x);
  unsafe{
    println!("allocated value is: {}",
    *raw_pointer);
  }
}
```

Unsafe Superpowers

Unsafe Rust
≠
Vulnerable Code

C2Rust Refactoring Tool

This is a refactoring tool for Rust programs, aimed at removing unsafety from automatically-generated Rust code.

https://c2rust.com/manual/c2rust-refactor/

Ownership Analysis

Method:

1. Permission Levels: READ < WRITE < MOVE
2. Permission level is assigned to every use of a pointer
3. Does it comply with Rust principles?

Problems:

- Not finished
- Not updated in 4 years

```
fn f /* <s0, s1> */ (arr: /* s0 */ *mut Array) -> /* s1 */ *mut i32
        /* where s1 <= s0 */ {
    g(arr)
}

fn g /* <s0, s1> */ (arr: /* s0 */ *mut Array) -> /* s1 */ *mut i32
        /* where s1 <= s0 */ {
    ...
}
```

In Rust We Trust – A Transpiler from Unsafe C to Safer Rust

Michael Ling[1], Yijun Yu[1,2], Haitao Wu[1], Yuan Wang[1], James R. Cordy[3], Ahmed E. Hassan[3]
[1] Huawei Technologies, Canada; [2] The Open University, UK; [3]Queen's University, Canada

Translating C to Safer Rust

MEHMET EMRE, University of California Santa Barbara, USA
RYAN SCHROEDER, University of California Santa Barbara, USA
KYLE DEWEY, California State University Northridge, USA
BEN HARDEKOPF, University of California Santa Barbara, USA

FROM C

TOWARDS IDIOMATIC & SAFER RUST

THROUGH CONSTRAINTS-GUIDED REFACTORING

TAN YAO HONG, BRYAN

Strategies

Ownership Analysis

- C2Rust Refactoring Tool

Optimistic Rewriting

- Translating C to Safer Rust

Constraint Solving

- From C Towards Idiomatic & Safer Rust
- In Rust We Trust

Michael Ling, Yijun Yu, Haitao Wu, Yuan Wang, James R. Cordy, and Ahmed E. Hassan. 2022. In rust we trust: a transpiler from unsafe C to safer rust. In Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (ICSE '22). Association for Computing Machinery, New York, NY, USA, 354–355. https://doi.org/10.1145/3510454.3528640

Optimistic rewrite

Method:

1. Do optimistic rewrite
   Eg: change all raw pointers to borrowed references
2. Iteratively fix compiler errors

```
pub unsafe fn find(value: c_int, node: *mut node_t) -> *mut node_t
```

```
1 fn find<'a1, 'a2, 'a3, 'a4, 'a5, 'a6>(value: c_int, mut node: Option<&'a1 mut node_t<'a2, 'a3>>) ->
2     Option<&'a4 mut node_t<'a5, 'a6>>;
```

Optimistic rewrite

```
1  pub fn find<'a1, 'a2, 'a3, 'a4, 'a5, 'a6>(mut value: c_int, mut node: Option<&'a1 mut node_t<'a2, 'a3>>)
2   -> Option<&'a4 mut node_t<'a5, 'a6>>
3  where 'a1: 'a4, 'a5: 'a2, 'a6: 'a3, 'a3: 'a6, 'a2: 'a5
4  {
5      if value < **(**node.as_ref().unwrap()).value.as_ref().unwrap() && !(**node.as_ref().unwrap()).left.
         is_none() {
6          return find(value, borrow_mut(&mut (*node.unwrap()).left))
7      } else {
8          if value > **(**node.as_ref().unwrap()).value.as_ref().unwrap() && !(**node.as_ref().unwrap()).
         right.is_none() {
9              return find(value, borrow_mut(&mut (*node.unwrap()).right))
10         } else { if value == **(**node.as_mut().unwrap()).value.as_mut().unwrap() { return node } }
11     }
12     return None;
13 }
```

Problems:

- Non-idiomatic code
- Programmer's intent is in danger

Mehmet Emre, Ryan Schroeder, Kyle Dewey, and Ben Hardekopf. 2021. Translating C to safer Rust. Proc. ACM Program. Lang. 5, OOPSLA, Article 121 (October 2021), 29 pages. https://doi.org/10.1145/3485498

Sergey, I. 2022. *TAN YAO HONG, BRYAN*.FROM C TOWARDS IDIOMATIC & SAFER RUST THROUGH CONSTRAINTS-GUIDED REFACTORING.

Constraint Solving

$$Offset(t_1, t_2, t_3), Deref(t_3, t_4) \iff Index(t_1, t_2, t_4)$$

Method:

1. Create constraints based on AST nodes
   Eg: Offset(α, α_ind, α_out)
2. Solve constraints based on rules
3. Rewrite code based on solved constraints

From C Towards Idiomatic & Safer Rust: KU Leuven's CHR (Constraints Handling rules) system

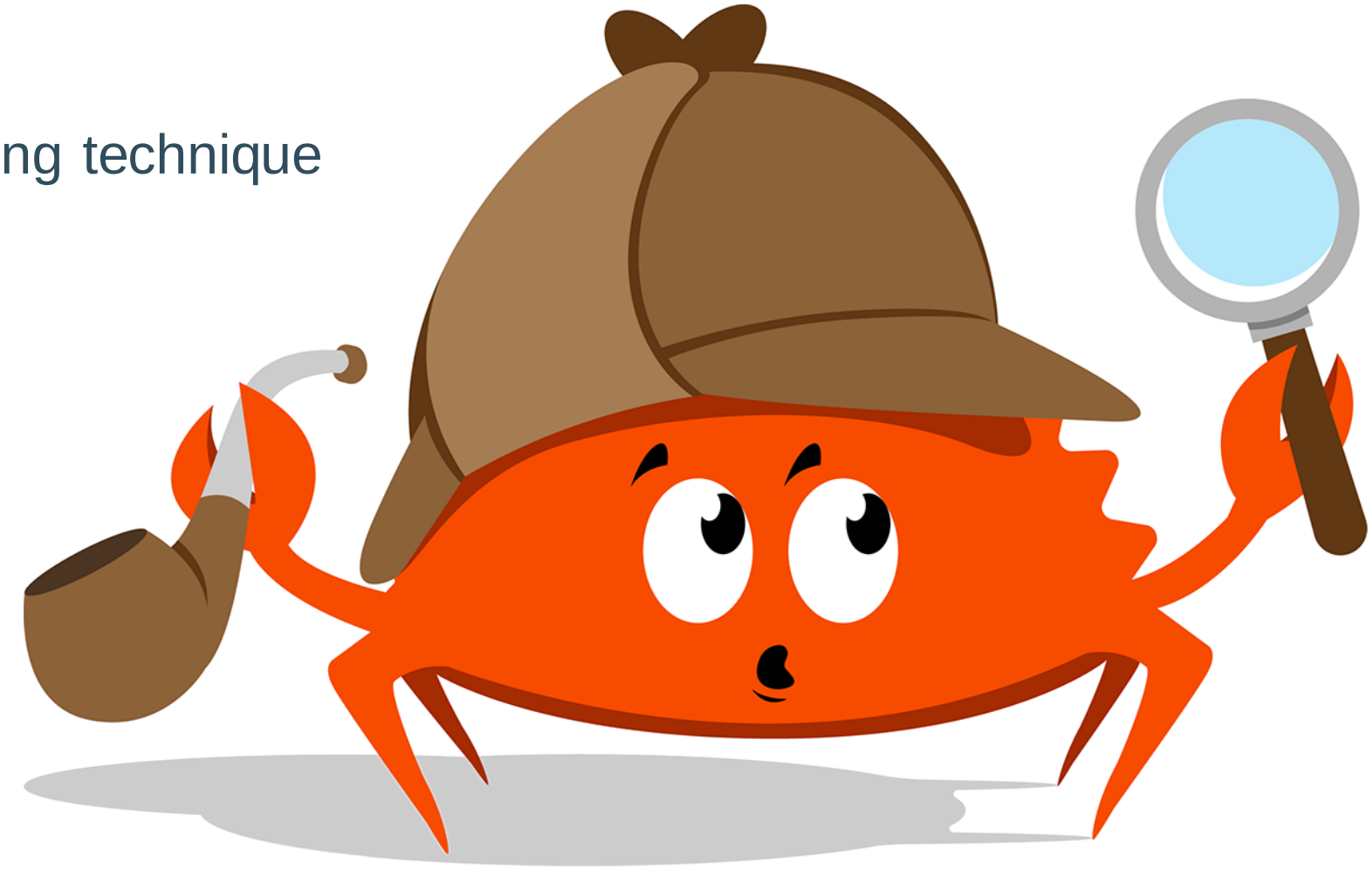
In Rust We Trust: TXL

Problems:

- Evaluation is minimal

Research Plans

Build on constraint solving technique



Conclusion

- C2Rust can only translate syntax
- Patience

Questions?