RUST GLOBAL × 

RustChinaConf 2025&
Rust Global China

Improving the incremental system in the Rust compiler

Alejandra González

你好!

About me

01 Clippy team member (Rust's linter)

02 Obsessed with performance

03 Rust Project Goal owner twice
(40-60% faster than last year)

04 Google OSPB Award winner

CHAPTER 0

THE PROBLEM

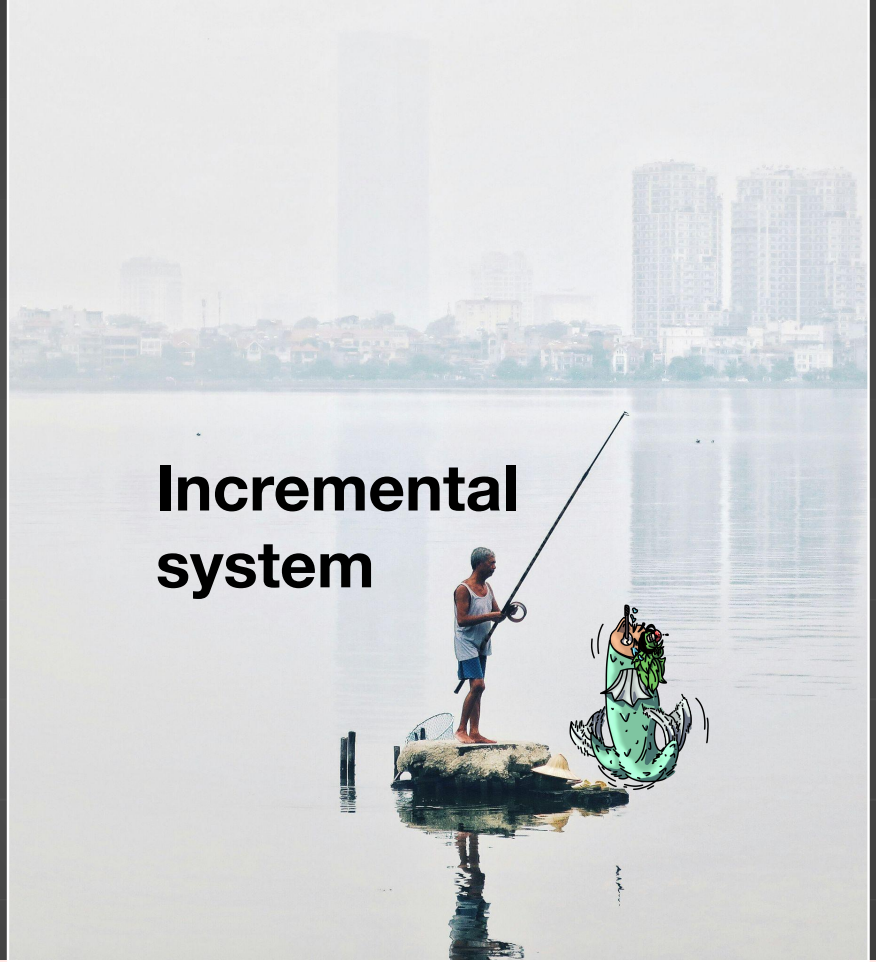
The problem

```
//...
Compiling bevy_winit v0.16.1
Compiling bevy_ui v0.16.1
Compiling bevy_audio v0.16.1
Compiling bevy_gilrs v0.16.1
Compiling bevy_internal v0.16.1
Compiling bevy v0.16.1
Compiling test-crate5 v0.1.0 (/home/meow/git/test-crate5)
Finished `dev` profile [unoptimized + debuginfo] target(s) in 4m 28s

~>
```


The problem

Visual representation of me being haunted by these complex technical issues

The problem



The problem

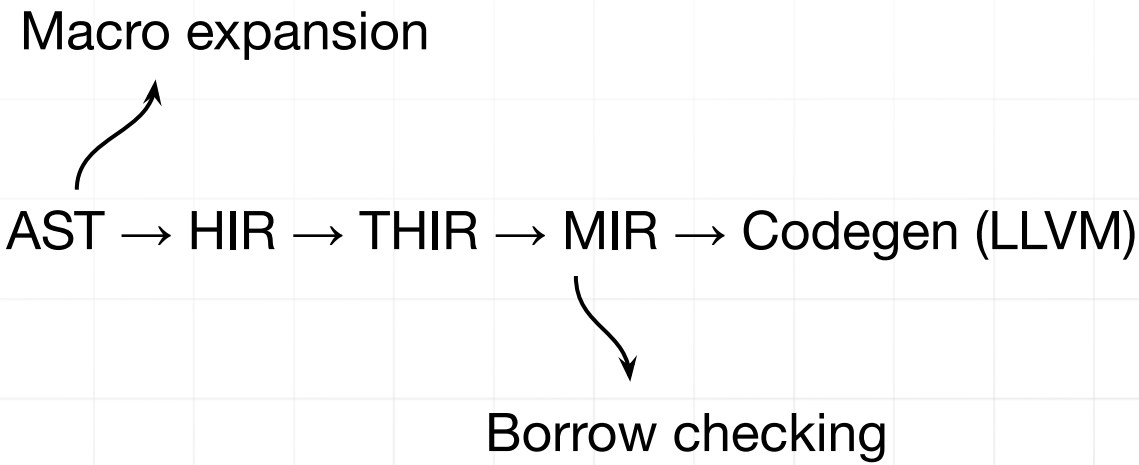
```
~> cargo build
   //...
   Compiling bevy_state v0.16.1
   Compiling bevy_gizmos v0.16.1
   Compiling bevy_winit v0.16.1
   Compiling bevy_ui v0.16.1
   Compiling bevy_audio v0.16.1
   Compiling bevy_gilrs v0.16.1
   Compiling bevy_internal v0.16.1
   Compiling bevy v0.16.1
   Compiling test-crate5 v0.1.0 (/home/meow/git/test-crate5)
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 4m 29s

~> cargo check
   //...
   Compiling bevy v0.16.1
   Compiling test-crate5 v0.1.0 (/home/meow/git/test-crate5)
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 1m 57s
   // TARGET IS NOT REUTILIZED!!                                  ^^^^^^
```

Target size
=
Build + Check

Borrow checking, trait solving, macro expansion… THRICE!!

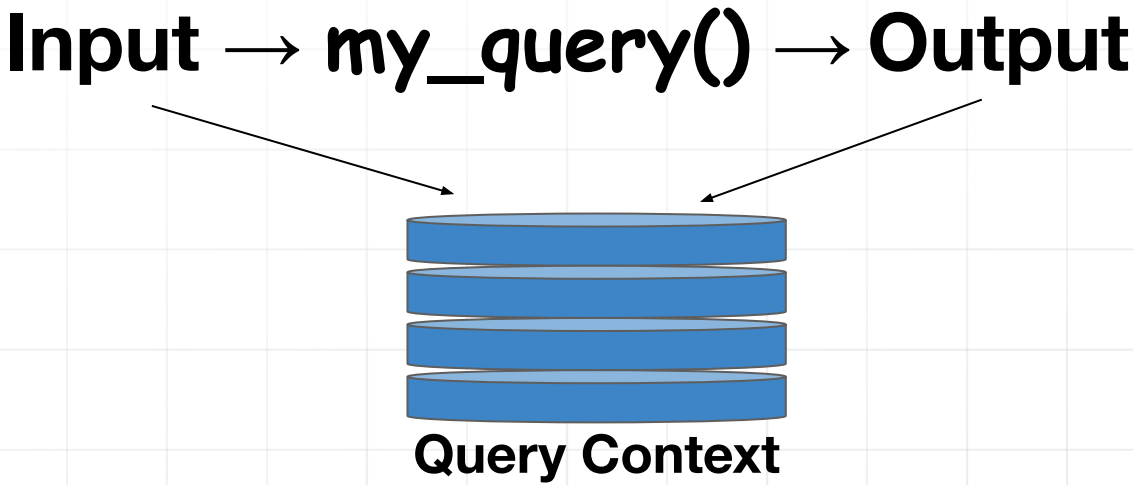
(That’s not good)

How the compiler works

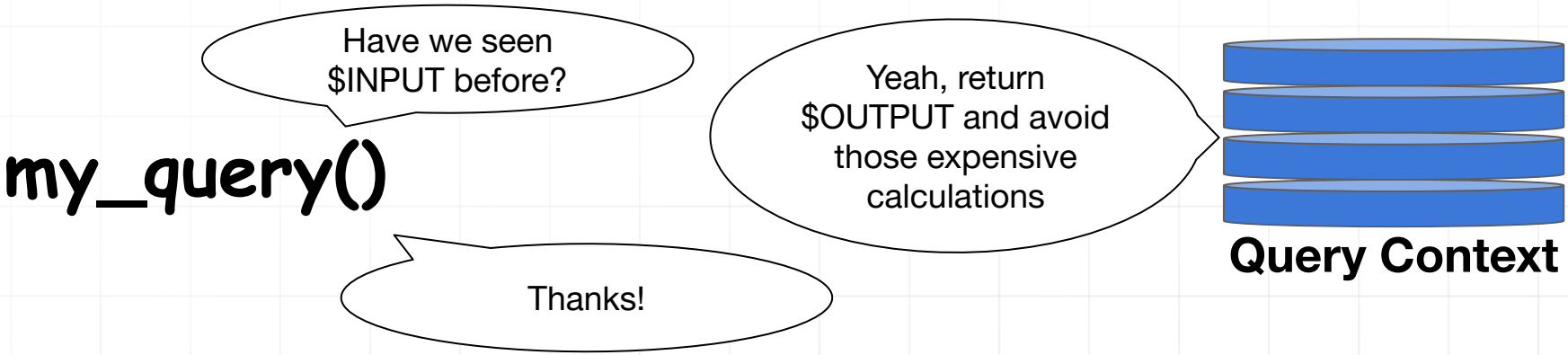


How the compiler works

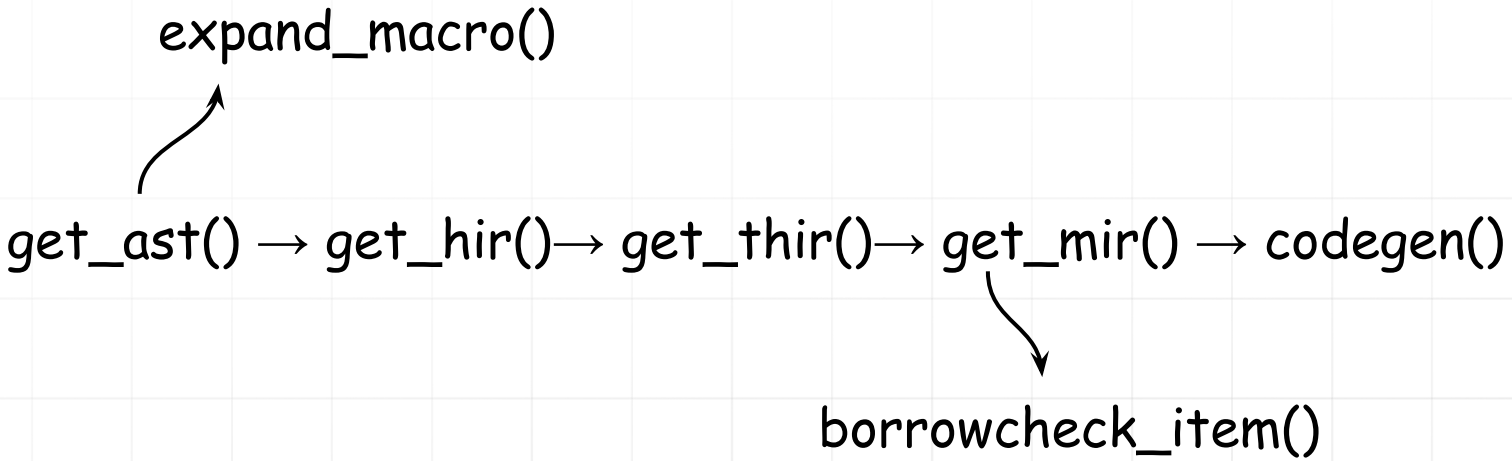
How the compiler works



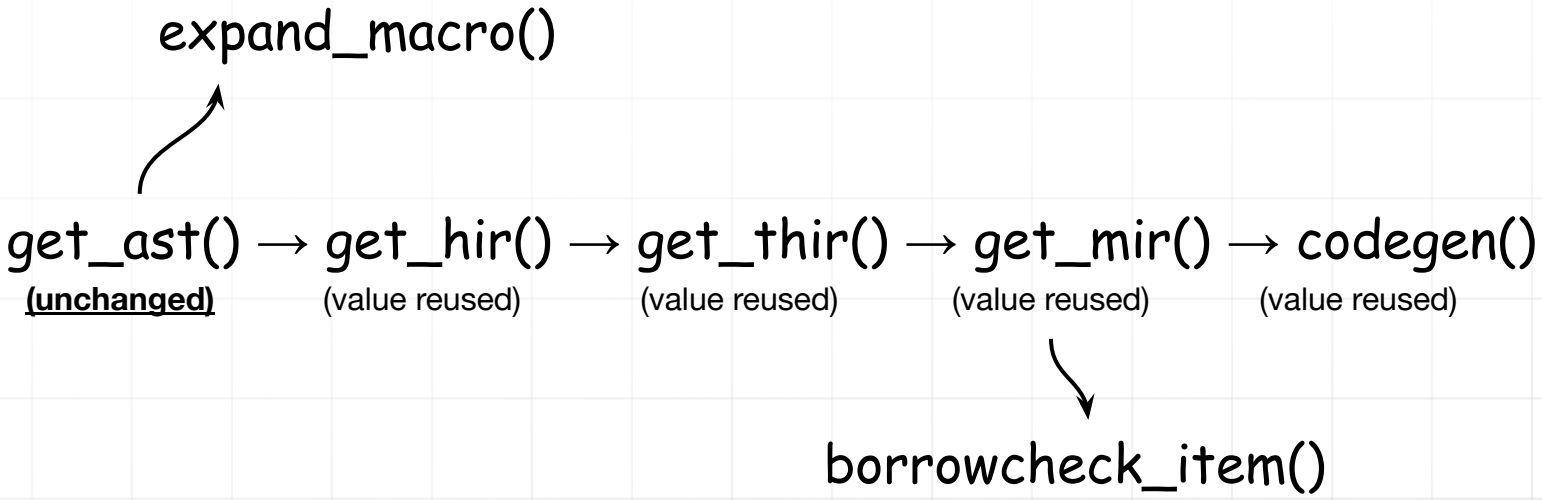
How the compiler works



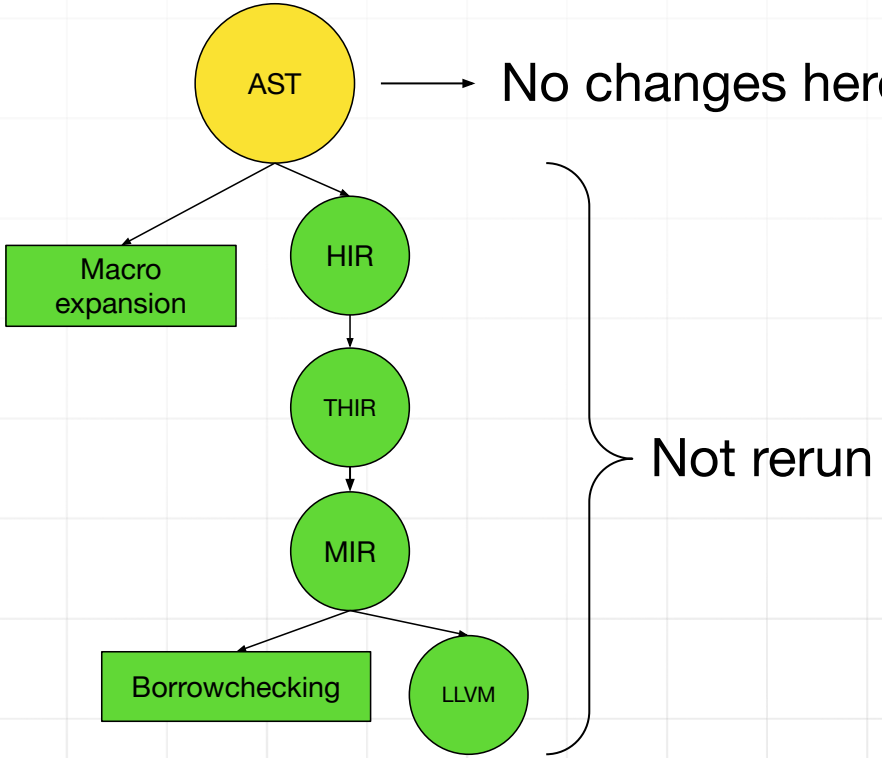
How the compiler works



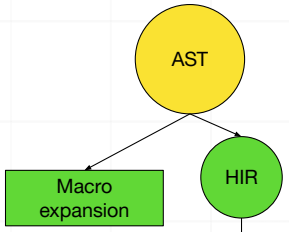
How the compiler works



How the compiler works



How the compiler works



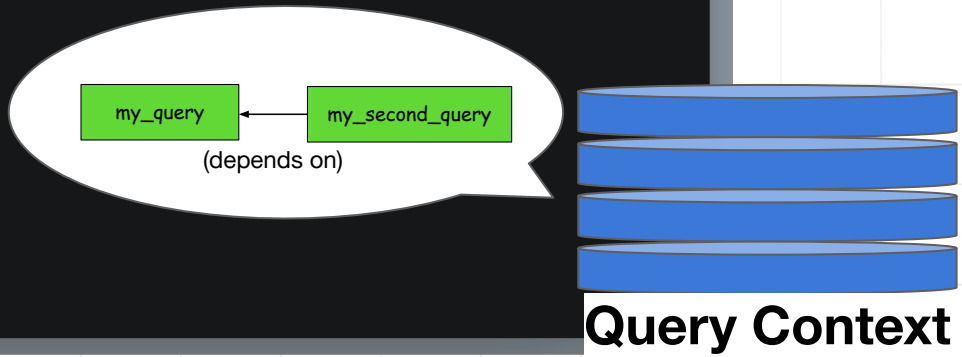
save

nes?

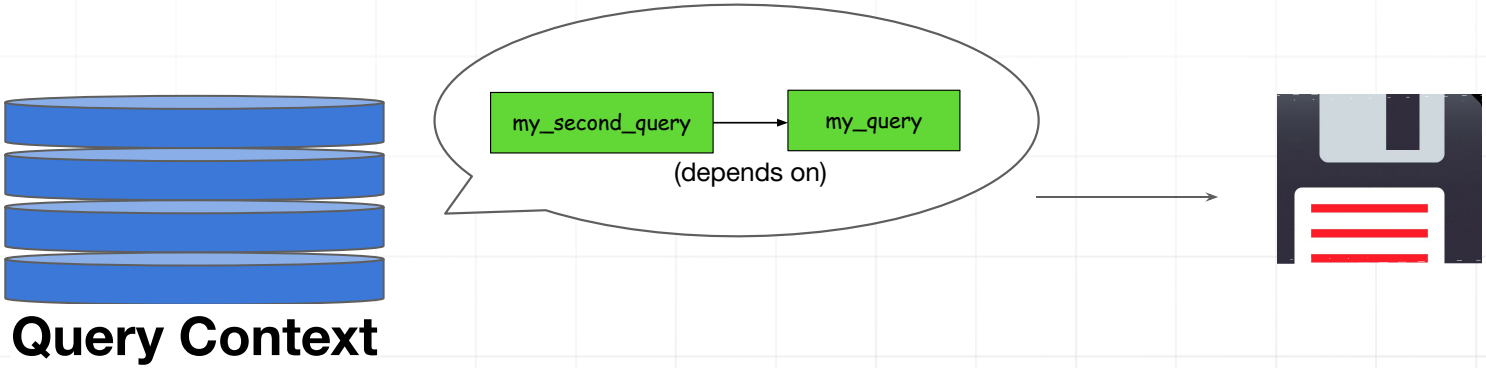
SIMPLIFICATION

How the compiler works

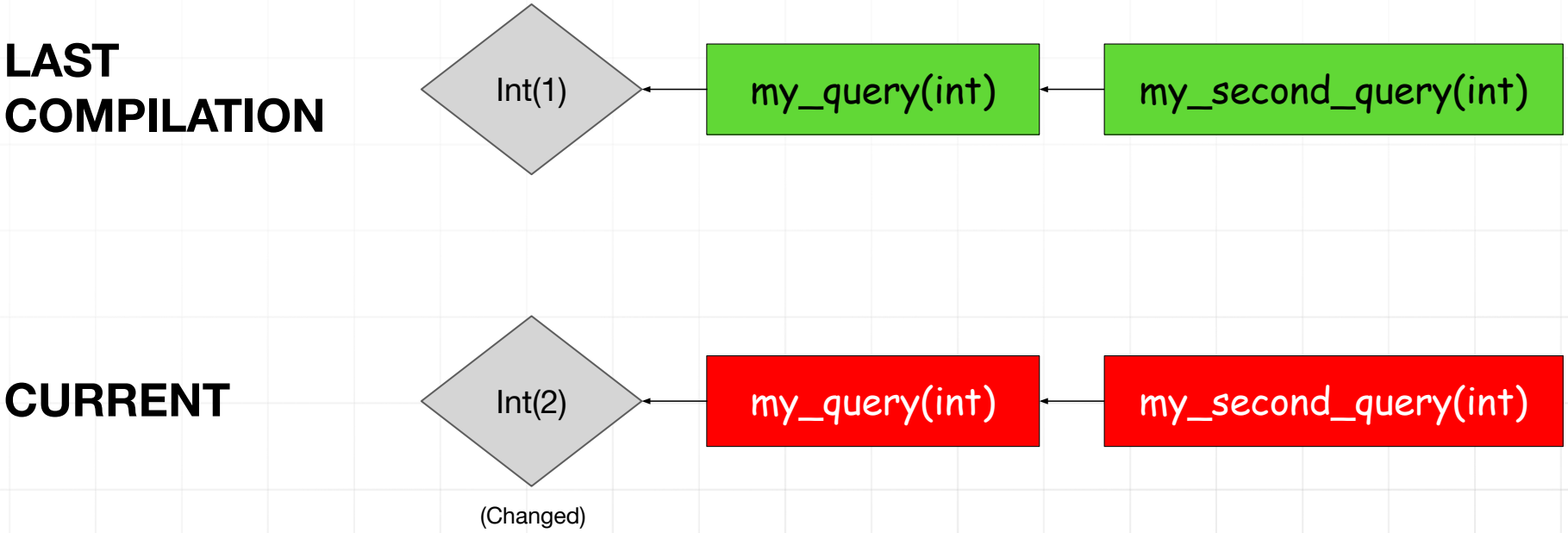
```rust
fn my_query(tcx: TyCtxt /* Typing context */, key: usize) {
  if key % 2 == 0 {
    // `my_second_query` gets saved to the query context
    // as dependant on `my_query`
    tcx.my_second_query(key)
  } else {
    unreachable!();
  }
}
```



How the compiler works



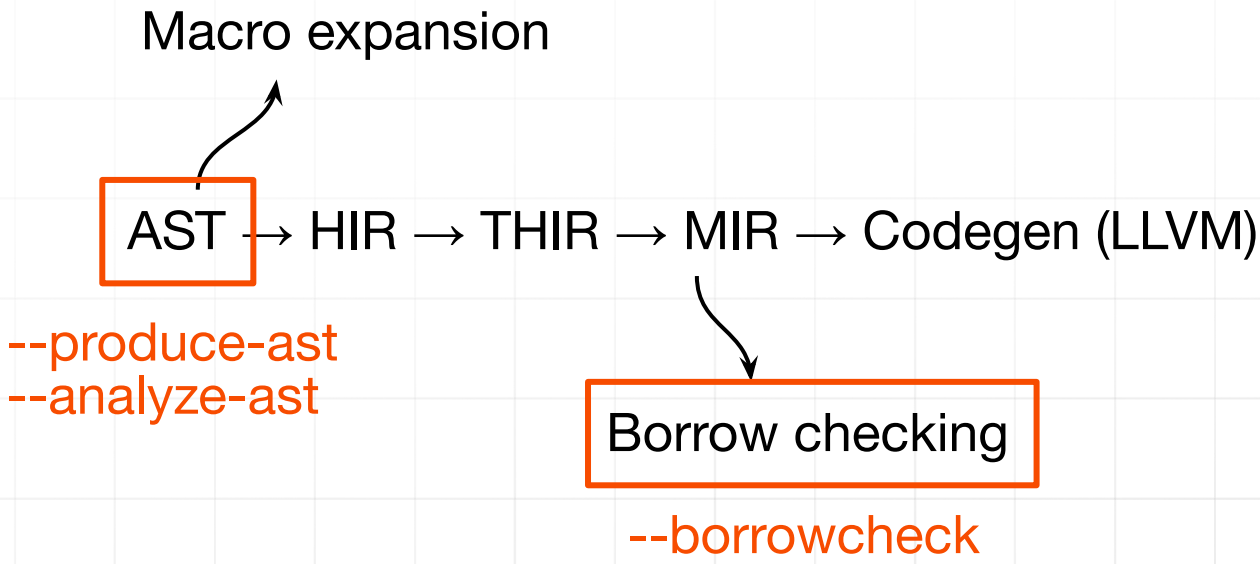
How the compiler works



CHAPTER 1

ATOMIC BITS
& DATA DEPENDENCIES

Atomic bits


Macro expansion
AST → HIR → THIR → MIR → Codegen (LLVM)
--produce-ast
--analyze-ast
Borrow checking
--borrowcheck

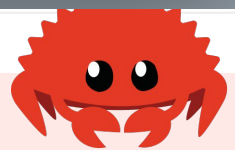
Atomic bits

```
CARGO=/home/meow/.rustup/toolchains/nightly-2025-07-15-x86_64-unknown-
linux-gnu/bin/cargo rustc --crate-name dep_d --edition=2024 dep_d/src/
lib.rs --error-format=json  --crate-type lib --emit=dep-info,metadata -C
embed-bitcode=no -C debuginfo=2 --atomic-bits="borrowcheck" --check-cfg
'cfg(docsrs,test)' --check-cfg 'cfg(feature, values())' -C
metadata=36baaed7c489dc73 -C incremental=/home/meow/git/test-crate5/
target/debug/incremental -L dependency=/home/meow/git/test-crate5/target/
debug/deps
```

Atomic bits

```
cargo clippy
cargo clippy --atomic-bit="borrowcheck"
```

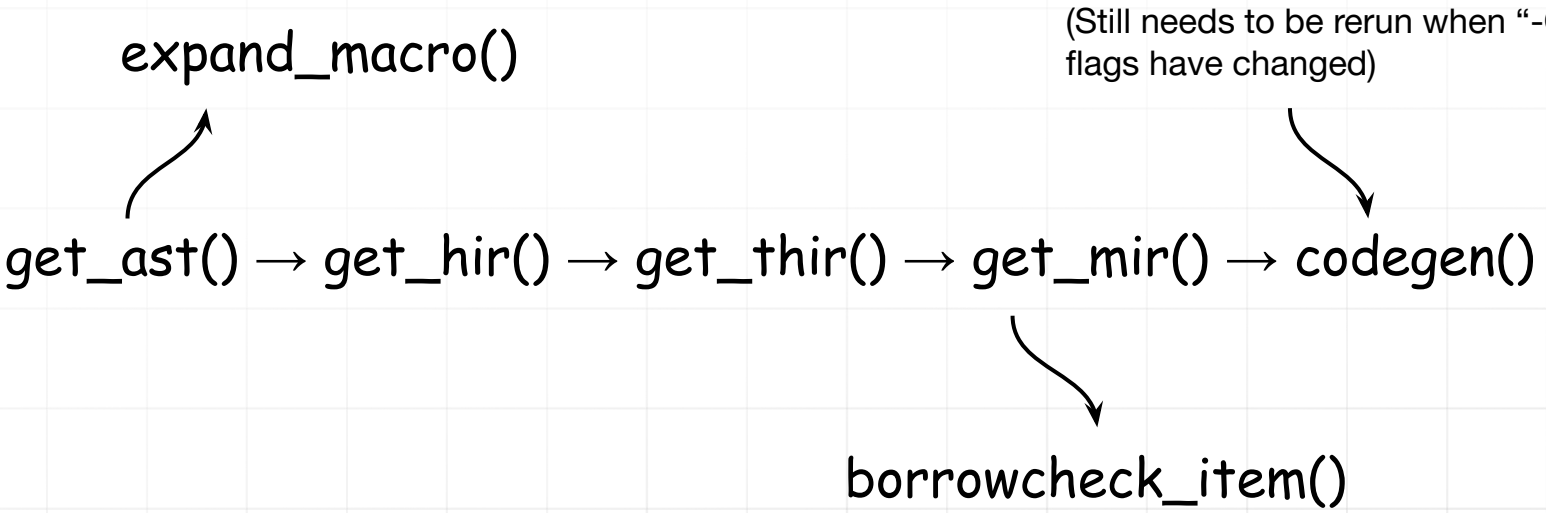
```
cargo build
cargo build --atomic-bit="emit-bin"
```



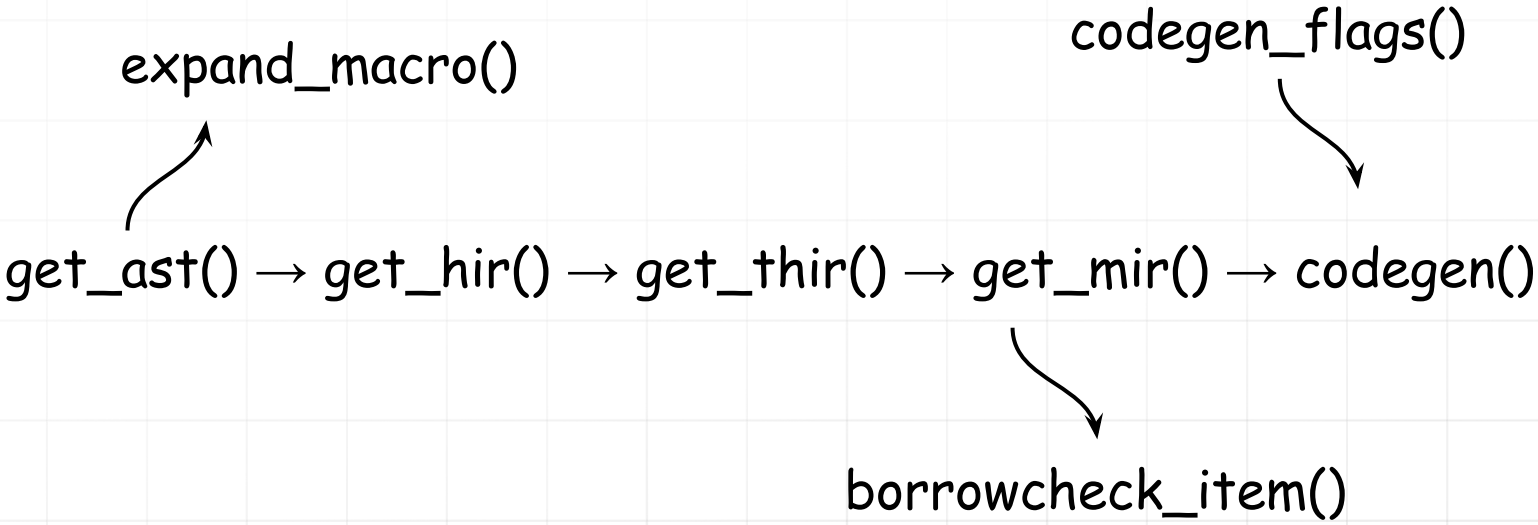
CHAPTER 1

ATOMIC BITS & DATA DEPENDENCIES

Data dependencies

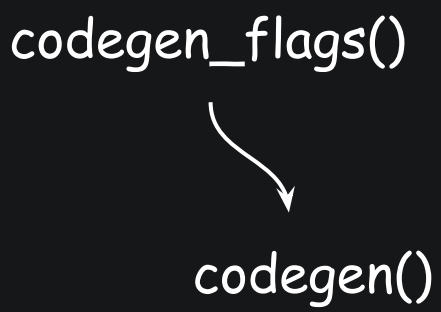


Data dependencies

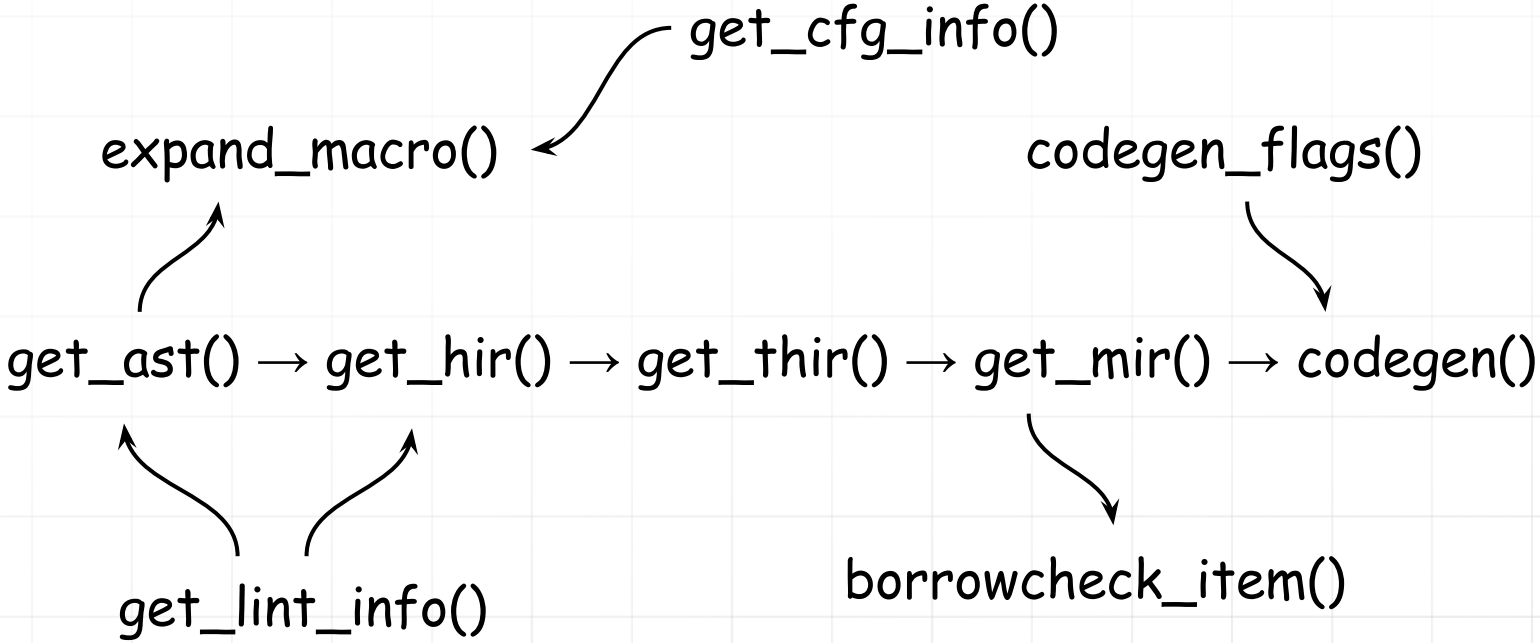


Data dependencies

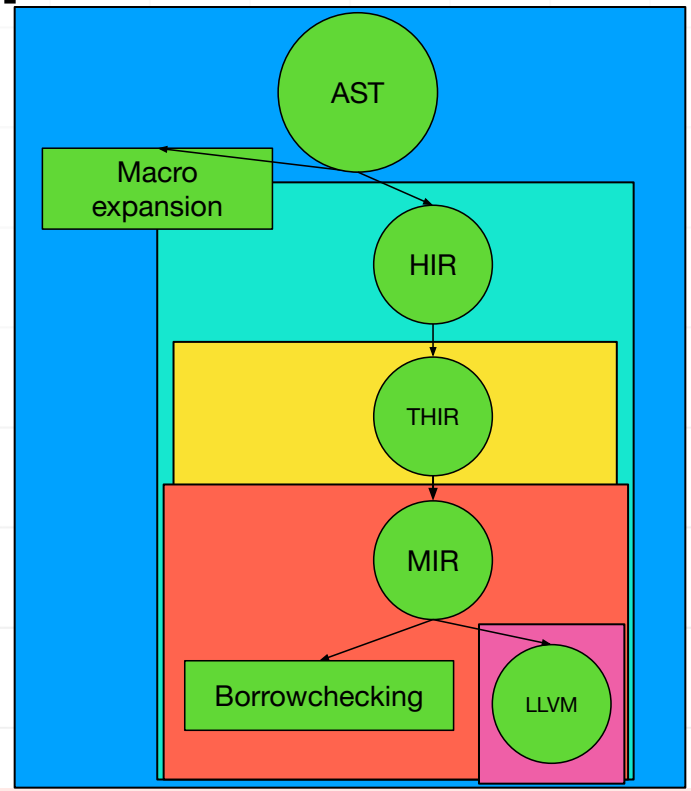
```rust
fn codegen_flags(tcx: TyCtxt, cli_input: String) -> CodegenFlags {
  Codegen_flags {
    linker: ..
    linker_args: ..
    lto: ..
    loop_vectorization: ..
  }
}
```



Data dependencies

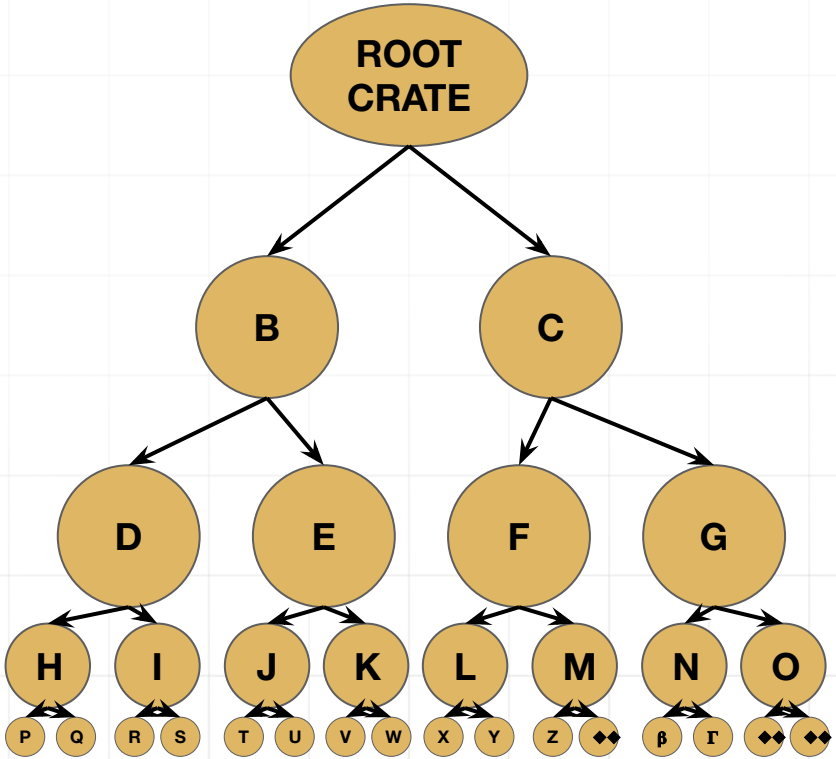


How the compiler works



THE END?

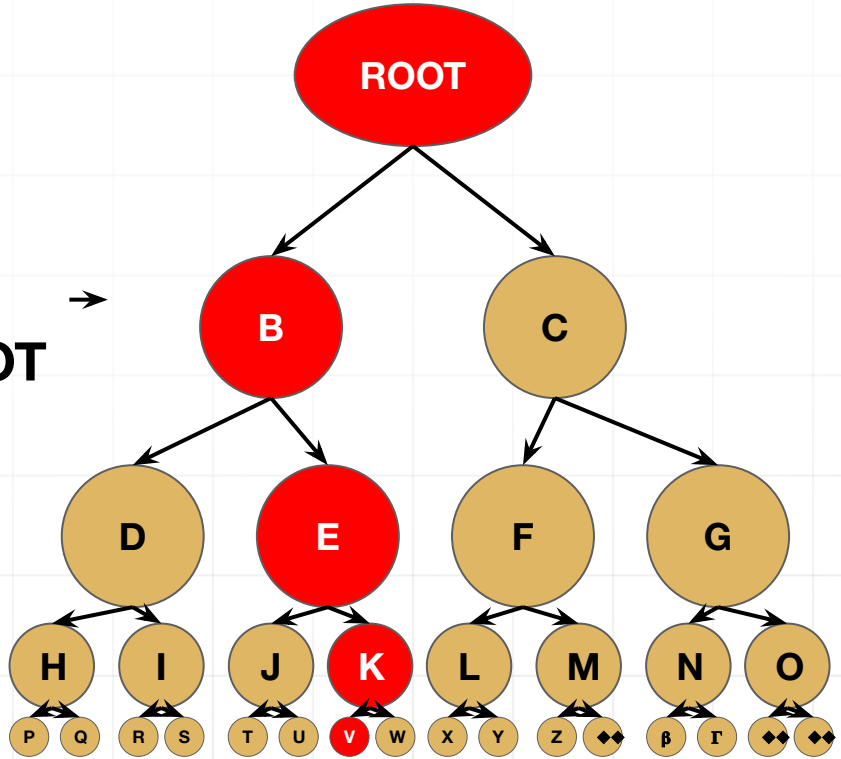
```
   // ...
   Compiling rustc_public v0.1.0-preview
   Compiling rustc_interface v0.0.0
   Compiling ctrlc v3.4.7
   Compiling rustc_log v0.0.0
   Compiling jiff v0.2.15
   Compiling rustc_driver_impl v0.0.0
   Compiling rustc-main v0.0.0
   Compiling rustc_driver v0.0.0
    Finished `release` profile [optimized + debuginfo] target(s) in 15m 52s

~> git diff
  compiler/rustc_data_structures/src/fx.rs --- Rust
  29 29      };
  30 30  }
   31
   32 #[allow(unused)]
   33 const A: u32 = 0; // This is unused
```


ROOT
CRATE
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
◆◆
β
Γ
◆◆
◆◆

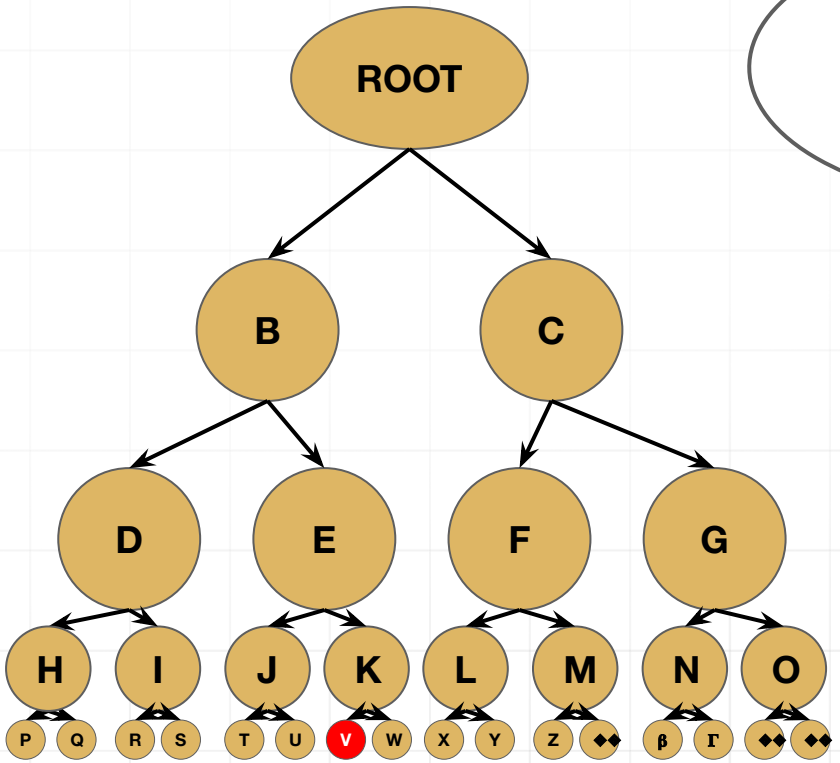
Rebuild time

=

V + K + E + B + ROOT



Only changed a comment in V



V HAS CHANGED!!!
Rebuilding the tree

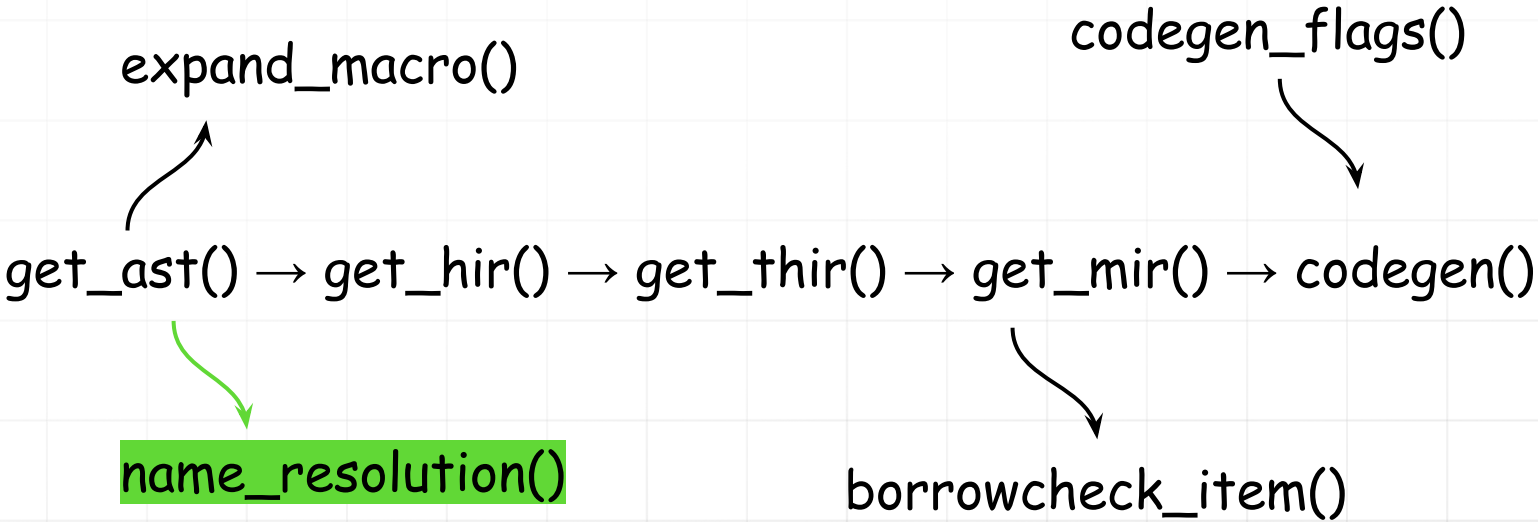
Things that Cargo checks:

Compiler version	Target name
Enabled features	Last modified time of sources
LTO flags	Immediate dependency's hashes
Compile mode (tests, build, doc…)	…etc

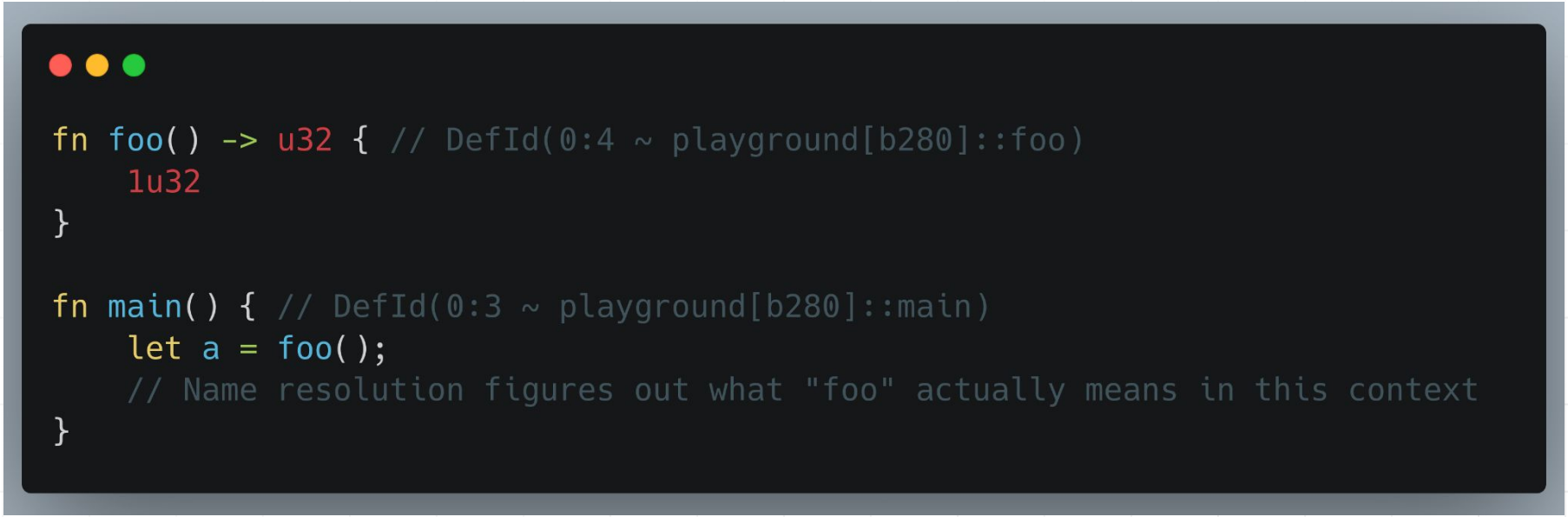
CHAPTER 3

TWO-STAGE FINGERPRINTS

Two-stage fingerprints



Two-stage fingerprints



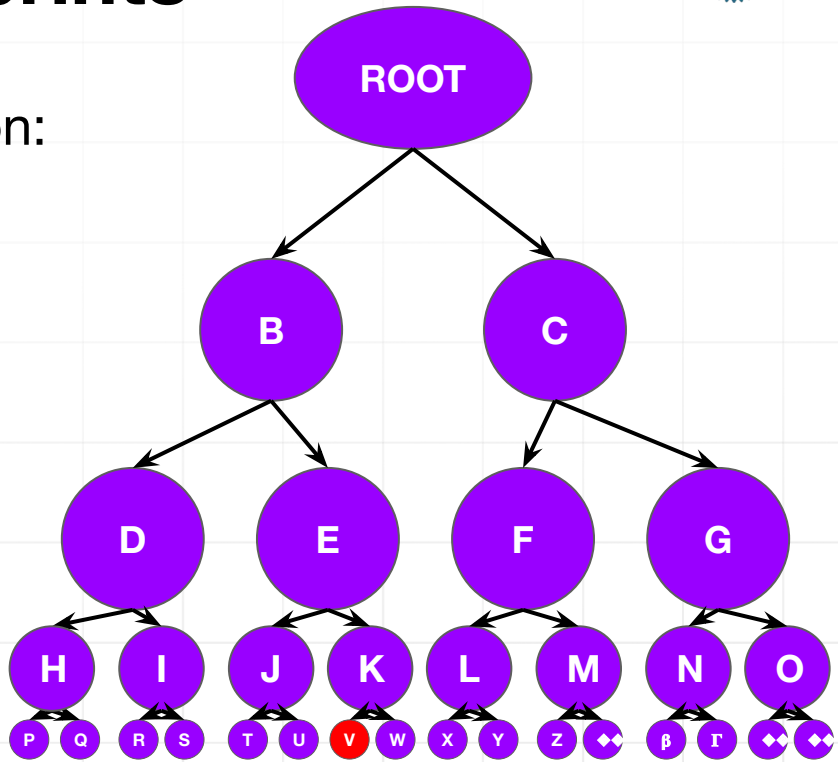
```rust
fn foo() -> u32 { // DefId(0:4 ~ playground[b280]::foo)
    1u32
}

fn main() { // DefId(0:3 ~ playground[b280]::main)
    let a = foo();
    // Name resolution figures out what "foo" actually means in this context
}
```

Two-stage fingerprints

Generated on name resolution:

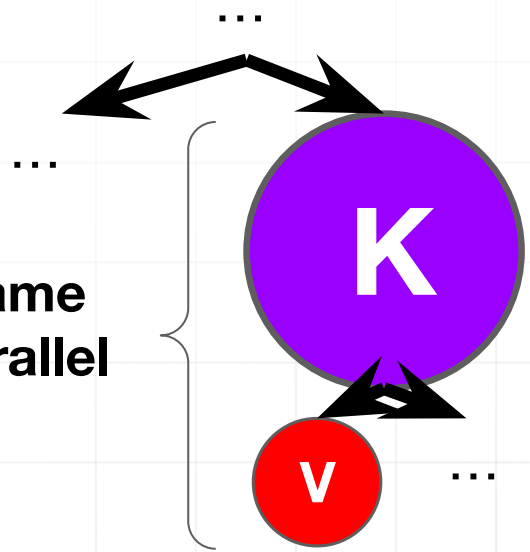
- `DefId(0:4 ~ playground[b280]::foo)`
- `DefId(0:3 ~ playground[b280]::main)`

Does K depend on foo's **behaviour?**

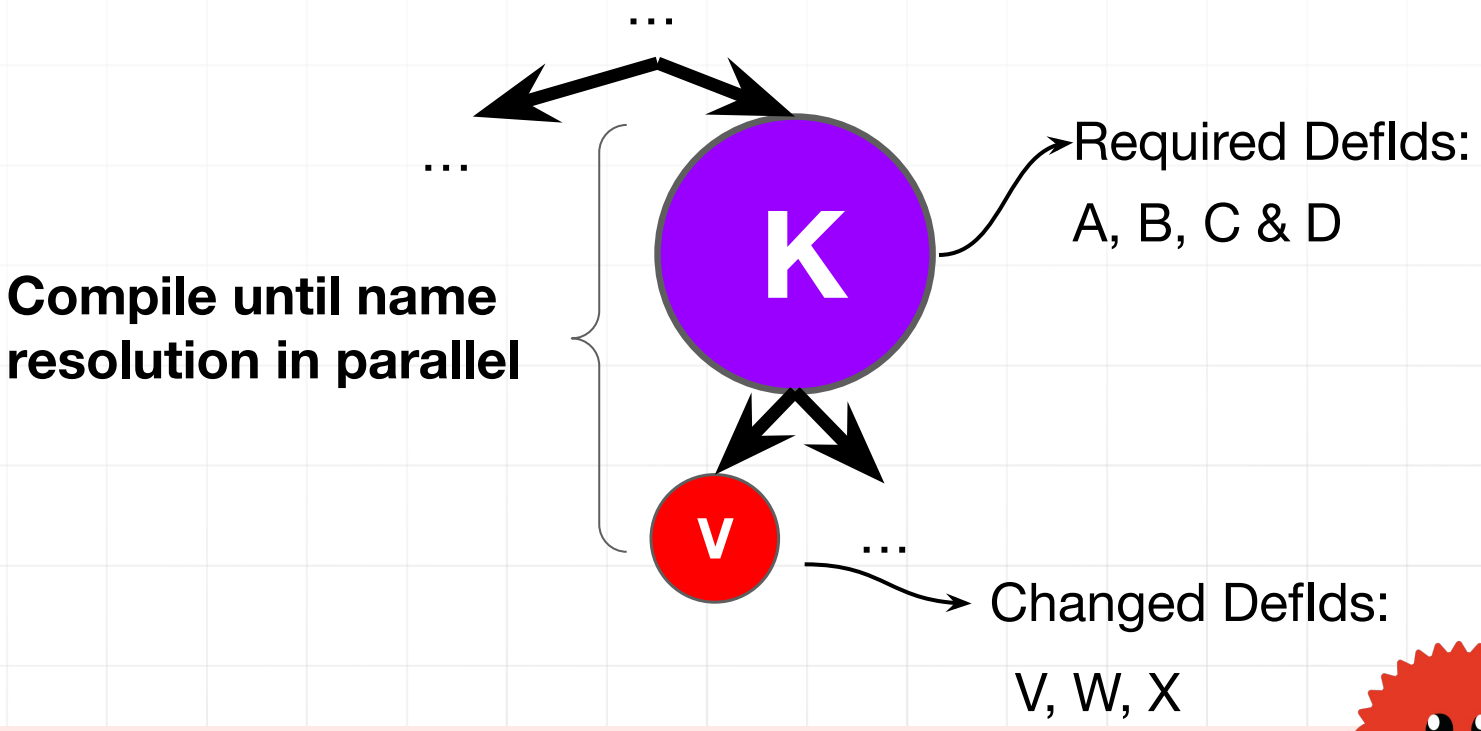


Way 1

Compile until name resolution in parallel



Way 1


...
...
K
Required DefIds:
A, B, C & D
Compile until name
resolution in parallel
V
...
Changed DefIds:
V, W, X

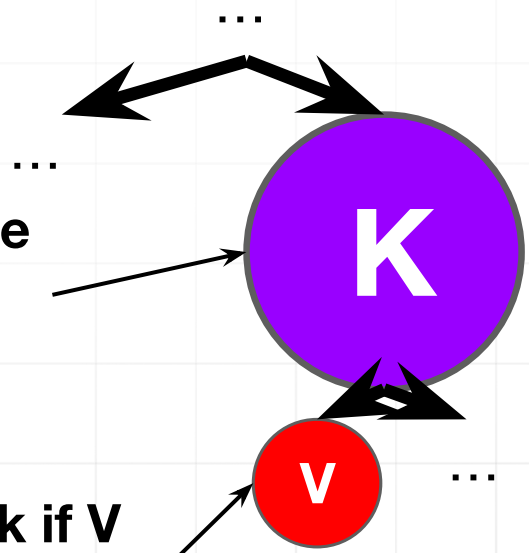
Way 1

Dependency	Dependant
Required DefIds: A, B, C & D	Changed DefIds: V, W, X

NO OVERLAP?

NO RECOMPILATION

Way 2

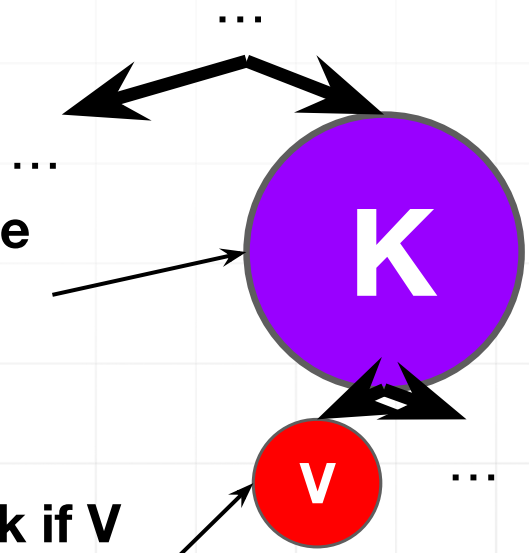


After compilation, save the used symbols to incremental

In the next one, check if V changes one that relates to those used in K

We don’t have to even parse K!

Way 2



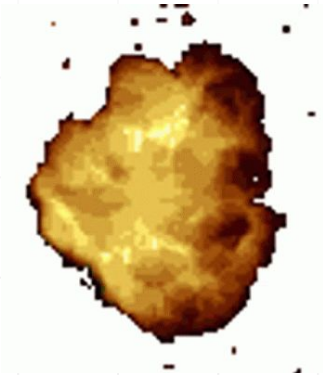
After compilation, save the used symbols to incremental

In the next one, check if V changes one that relates to those used in K

We don’t have to even parse K!

How will we inform Cargo?

WITH ATOMIC BITS



The end

Talk by Alejandra González

Illustrations by @lux_mangastyle

RUST GLOBAL ×

RustChinaConf 2025&
Rust Global China

Thank You!