Charon: Seeing What rustc Sees

Guillaume Boisseau aka Nadrieril (Inria Paris),

With *Son Ho, Opale Sjöstedt, Aymeric Fromherz, and contributors*

Hi I’m Nadri

- Background: academia (programming languages)

- Rust contributor since 2019
    - I maintain pattern-matching
    - Active on lang design, dabble in types & opsem & spec

- Employed on the Charon/Aeneas project
    - Translates Rust to Lean for formal verification

It’s hard to know what rustc sees

- Rust leaves a lot implicit
    - Traits, type inference, names, const eval, drops, layouts, generated code, …

- Only sure way is to hook into rustc
    - Means wrangling compiler-internal APIs
    - A lot is still implicit/scattered

Charon shows you what rustc sees

-> A common core for analysis tools

Demo time!

Demos

- Demo 1: Inspect kernel crate and layouts
- Demo 2: Analyze call-graph
- Demo 3: Transpile to Python

Charon saves you work

- Everything explicit (types, coercions, method calls, vtables, …)
- Real layouts and (soon) Reference-guaranteed layouts
- Choose polymorphic vs monomorphic output
- Choose control-flow graph vs normal match/loop blocks
- Can evaluate constants to raw bytes
- Drop flags
- Multi-target support
- Faster postcard format

Charon today

- 4 years of expertise
- Supports all of Rust except async
- Tested against Miri (thanks to Soteria Rust)
- Beta version, working on 1.0
    - Long tail of bugs
    - Future-proof the representation

Conclusion

- Charon: A common core for Rust tools
  - Shows you what rustc sees
  - Easy to consume
  - Try it out!

https://github.com/AeneasVerif/charon

Demo repo:
https://github.com/AeneasVerif/kangrejos-2026-charon-demo

Zulip:
https://aeneas-verif.zulipchat.com/