Review

- `send` trait introduces an “isolation boundary” between threads
    - Objects can only be *copied* or *moved* between them

- **sync** trait tells the compiler an object is data-race free
  - And is implicitly **send**

- These traits need to be checked recursively for all members
  - *C++23 can not do this*
  - C++26 reflection can

- Lifetime safety is inherently intertwined with thread safety
    - *Solved in other languages with borrow checking, reference counting or mutable value semantics*

- We need to encapsulate pointers in value types to ensure they're not exposed to abuse
    - C++26 reflection generation (and future metaclasses) can make this simple

89

Review

- `send` trait introduces an "isolation boundary" between threads
  - Objects can only be ***copied*** or ***moved*** between them
- `sync` trait tells the compiler an object is data-race free
  - And is implicitly `send`
- These traits need to be checked recursively for all members
  - *C++23 can not do this*
  - C++26 reflection can
- Lifetime safety is inherently intertwined with thread safety
  - *Solved in other languages with borrow checking, reference counting or mutable value semantics*
- We need to encapsulate pointers in value types to ensure they're not exposed to abuse
  - C++26 reflection generation (and future metaclasses) can make this simple

Limitations