Inference

Inference occurs when the compiler can deduce certain semantic info without it being explicitly specified in the source code. Inference applies to types, impls and generic arguments. Places where inference is enabled: * Type inference onlet * missing statement type clauses. * Impl inference on method calls. * Impl inference when calling by trait. * Generic argument inference. Examples: let x = 1_u8; // x is inferred to be u8 let y = Option<_>::Some(x); // _ is inferred as u8, and y is inferred as Option let u: u32 = Into::::into(x); // concrete impl for Into is inferred let u: u32 = x.into(); // The generic arguments for Into<?,?> and the concrete impl // are inferred

Complex impl example

trait Display { fn display(x: T) -> Array; } impl IntoArrayU8Display>> of Display { fn display(x: T) -> Array { x.into() // Inferred as TIntoArrayU8::into(x) } } impl U32IntoArrayU8 of Into> { fn into(x: u32) -> Array { // ... } } fn main() { let x = 1_u32; let y = Display::display(x); // Inferred as IntoArrayU8Display::display(x) }

Search locations for impl

When inferring an impl of a trait, the compiler will search for the impl in the following locations: * Defined generic parameters in the current scope. * The current module. * The core module. * Defining module of the trait. * Defining module of each of the generic arguments of the trait. For example, the impl forDisplay will be searched for in the module whereusize is defined.

Multiple impls and ambiguity

When searching for an impl, and multiple results are found for the same concrete trait (e.g.Display), the compiler will throw an error. This is to prevent ambiguity in the code. In this case, the user must specify the impl name explicitly.

9.3 Generics 10. Panic