Context Bounds

Edit this page on GitHub

A context bound is a shorthand for expressing the common pattern of a context parameter that depends on a type parameter. Using a context bound, the `maximum` function of the last section can be written like this:

```
def maximum[T: Ord](xs: List[T]): T = xs.reduceLeft(max)
```

A bound like `: Ord` on a type parameter `T` of a method or class indicates a context parameter `using Ord[T]`. The context parameter(s) generated from context bounds come last in the definition of the containing method or class. For instance,

```
def f[T: C1 : C2, U: C3](x: T)(using y: U, z: V): R
```

would expand to

```
def f[T, U](x: T)(using y: U, z: V)(using C1[T], C2[T], C3[U]): R
```

Context bounds can be combined with subtype bounds. If both are present, subtype bounds come first, e.g.

```
def g[T <: B : C](x: T): R = ...
```

Migration

To ease migration, context bounds in Dotty map in Scala 3.0 to old-style implicit parameters for which arguments can be passed either with a `(using ...)` clause or with a normal application. From Scala 3.1 on, they will map to context parameters instead, as is described above.

If the source version is `future-migration`, any pairing of an evidence context parameter stemming from a context bound with a normal argument will give a migration warning. The warning indicates that a `(using ...)` clause is needed instead. The rewrite can be done automatically under `-rewrite`.

Syntax

```
TypeParamBounds   ::=  [SubtypeBounds] {ContextBound}
ContextBound      ::=  ':' Type
```

‹ Using ... Importi... ›

Contributors to this page

pikinier20 BarkingBad julienrf performantdata
odersky michelou ShapelessCat