Determina detalladamente cuál es el tipo de la función `f` definida por la ecuación:

```
f x y z = if (\x -> y > x) z then x y else (\y -> y.(+2)) x z.
```

El tipo de las expresiones a los lados de la igualdad debe ser el mismo, digamos `a`:

```
f x y z::a
if (\x -> y > x) z then x y else (\y -> y.(+2)) x z::a
```

Analizamos la expresión de la izquierda. Se trata de aplicaciones sucesivas, luego podemos asegurar que `f` tiene tipo funcional con tres argumentos y resultado de tipo `a`, por tanto:

```
f::a1 -> a2 -> a3 -> a
x::a1
y::a2
z::a3
```

Analizamos la expresión de la derecha: `if (\x -> y > x) z then x y else (\y -> y.(+2)) x z::a`
Puesto que `if::Bool -> t -> t -> t`, deducimos:

```
(\x ->  y >  x) z::Bool
x y::a
(\y -> y.(+2)) x z::a
```

Entonces, como `(\x -> y > x) z::Bool` $\equiv$ `(y > z)::Bool` y sabiendo que `(>)::Ord t => t->t->Bool`, podemos deducir que `z` e `y` tienen que tener el mismo tipo, digamos `b`, y que este tiene que estar en la clase `Ord`. Entonces `y::Ord b => b`, `z::Ord b => b`.

Por otra parte:
`x y::a` implica que `x::c -> a`, `y::c`

De `(\y -> y.(+2)) x z::a` deducimos `((x.(+2)) z)::a`, por lo que haciendo uso del tipo de la composición de funciones `(.)` y del tipo de `(+)`, obtenemos `x::Num d => d -> a`, `z::Num d => z::d`

Juntándolo todo:
`x::a1`, `x::c -> a` , `x::Num d => d -> a`. Por tanto `c` = `d`, `a1` = `c -> a`, `Num c`
`y::a2`, `y::Ord b => b`, `y::c`. Por tanto `a2` = `c` = `b`
`z::a3`, `z::b`, `z::d` implica que `a3` = `b` = `d`

Por tanto, solo necesitamos la variable de tipo `a` y otra, digamos `b`, para `a2` = `a3` = `b` = `c` = `d` de la que sabemos que está en las clases `Num` y `Ord`. Sustituyendo en el valor inicial dado a `f`, obtenemos:

```
f::(Ord b, Num b) => (b -> a) -> b -> b -> a
```