Initial and terminal objects in a category are defined by universal construction

Definition 0.1. In a category $\mathcal{C}$:

- an object $\emptyset$ is called **initial** if for any object X in $\mathcal{C}$ there is a unique arrow $\emptyset \to X$
- an object 1 is called **terminal** if for any object X in $\mathcal{C}$ there is a unique arrow $X \to 1$

They are *dual* definitions: if A is initial in $\mathcal{C}$ then $\bar{A}$ is terminal in $\mathcal{C}^{op}$ and vice-versa.

Theorem 0.1. *The initial and terminal objects are unique up to isomorphism.*

Definition 0.2. If 1 is terminal in $\mathcal{C}$ we call arrows $1 \to X$ **elements**[1] of X

Here we have changed our point ov view: we turn the internal structure of an object into its external relationship with other objects

> It is as if morphisms connects to hidden features *inside* objects. They *reveal* the inner structure of objects.

(see the discussion in representable functors)

Example 0.1. *In* $\mathcal{S}et$

- *the initial object is the empty set* $\emptyset$
- *the terminal object is the singleton* $\{\star\}$

Example 0.2. *In the category* $\mathcal{P}$ *from a poset* $(P, \leq)$

- *the initial object is the* minimum
- *the terminal object is the* maximum

[1] or **constant of type X**

Example 0.3. *In $\mathcal{H}ask$ the initial object is the empty type* `Void` *and the unique arrow to any other type is the function*

```
absurd :: Void -> a
```

The terminal object is the singleton type `Unit` *and the unique arrow from any type is*

```
unit :: a -> Unit
unit _ = ()
```

See [1] Ch.5.1 and also Category Theory 4.1: Terminal and initial objects - YouTube.

[1] B. Milewski, *Category Theory for Programmers* (2019).