

# Natural Numbers, or the best way to enumerate anything

Marc Coiffier

## **Contents**

```
'utils require import
```

- Required module: [utils](#)

```
'Nat_context [ { Type '.Nat } { .Nat '.zero } { .Nat 'n -> .Nat ? '.succ } ] def
```

```
'Nat Nat_context { .Nat } prods "Natural" defconstr
```

```
'zero Nat_context { .zero } funs "0" defconstr
```

```
Nat 'n -> 'succ Nat_context { .succ ( n ( .Nat .zero .succ ) ) } funs "S n" defconstr !
```

The `Nat` type is defined to *Natural* .  $\lambda(n : \textit{Natural}).\mu(n)$  has type  $\forall(n : \textit{Natural}) (\text{Nat}^P : \textit{Natural} \rightarrow \textit{Set}_1), \text{Nat}^P 0 \rightarrow (\forall(n_0 : \textit{Natural}), \text{Nat}^P n_0 \rightarrow \text{Nat}^P (S n_0)) \rightarrow \text{Nat}^P n$  .

$\lambda(n : \textit{Natural}).S n$  has type  $\textit{Natural} \rightarrow \textit{Natural}$  .