

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

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