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 $Natural$. $\lambda(\mathrm{n} : Natural).\, \mu(\mathrm{n})$ has type $\forall(\mathrm{n} : Natural)\, (\mathrm{Nat}^P : Natural \to Set_1),\, \mathrm{Nat}^P\, 0 \to (\forall(\mathrm{n}_0 : Natural),\, \mathrm{Nat}^P\, \mathrm{n}_0 \to \mathrm{Nat}^P\, (S\, \mathrm{n}_0)) \to \mathrm{Nat}^P\, \mathrm{n}$.

$\lambda(\mathrm{n} : Natural).\, S\, \mathrm{n}$ has type $Natural \to Natural$.