lec 9

2024-11-11

Nat : Type

S : Nat → Nat

O : Nat

Bool : Type

False : Bool

True : Bool

Int : Type ?

MkInt : Nat → Int

Neg : Int → Int

ou

MkInt : Nat → Int (wrapper)

ou

MkInt : Nat → Nat → Int (que tal Sign? / que tal Bool?)

```
data Sign
  P : Sign
  Z : Sign
  N : Sign
```

ou

Z : Int

SP : Int → Int

SN : Int → Int

$3 \equiv SP\ (SP\ (SP\ Z))$

$-2 \equiv SN\ (SN\ Z)$

$0 \equiv Z$

$SN\ (SP\ (SP\ Z))$

Ints	Nats
0	MI O
1	MI 2
2	MI 4
3	MI 6
⋮	⋮
-1	MI 1
-2	MI 3
-3	MI 5
⋮	⋮

$\mathbb{N} \not\subseteq \mathbb{Z}$

especificação de inteiros

$(\mathbb{Z}; +, \cdot, -, 0, 1, Pos)$ + axiomas

$\mathbb{Z}_{\mathbb{N}} \overset{def}{=} \{0_{\mathbb{Z}}, 1_{\mathbb{Z}}, 2_{\mathbb{Z}}, \ldots\} = \{0\} \cup Pos$

Polimorfismo

```
idNat : Nat → Nat
idNat  n = n

idBool : Bool → Bool
idBool  b = b
```

```
id_{α : Type} : α → α
id x = x
```

isso define uma
α-indexada família de funções

("open-ended")

```
ListNat : Type
  Cons : Nat → ListNat → ListNat
  Nil  : ListNat
```

$$\frac{\text{List} : \text{Ty} \to \text{Ty} \qquad \alpha : \text{Ty}}{\text{List}\ \alpha : \text{Ty}}$$

```
data List α
  Nil  : List α
  Cons : α → List α → List α
```

```
length : List α → Nat
sum    : List Nat → Nat
```