Chapter 18: Natural Numbers in Agda

Zhenjiang Hu, Wei Zhang School of Computer Science, PKU October 28, 2022



Peano Natural Number

```
data \mathbb{N} : Set where zero : \mathbb{N} suc : \mathbb{N} \to \mathbb{N}
```

```
0 = zero
1= suc zero
2 = suc (suc zero)
3= suc (suc (suc zero))
```



Some Operations on Natural Numbers

```
_{-+}: \mathbb{N} \to \mathbb{N} \to \mathbb{N}

zero + n = n

suc m + n = suc (m + n)

_{-*}: \mathbb{N} \to \mathbb{N} \to \mathbb{N}

zero * n = zero

suc m * n = n + (m * n)

pred : \mathbb{N} \to \mathbb{N}

pred 0 = 0

pred (suc n) = n
```



Two Simple Theorems about Addition

```
0+: \forall (x: \mathbb{N}) \rightarrow 0 + x \equiv x

0+x = refl
```

```
+0 : \forall (x : \mathbb{N}) \rightarrow x + 0 \equiv x
+0 zero = refl
+0 (suc x) rewrite +0 x = refl
```



Associativity and Working with Holes

加法的结合律

```
+assoc : \forall (x y z : \mathbb{N}) \rightarrow x + (y + z) \equiv (x + y) + z
+assoc zero y z = refl
+assoc (suc x) y z rewrite +assoc x y z = refl
```

如何通过"hole"来交互式地开发以上的证明?

步骤1:通过?引入hole

```
+assoc zero y z = ?
```

Ctrl+c Ctrl+l

+assoc zero y $z = \{! \ 0!\}$



Associativity and Working with Holes

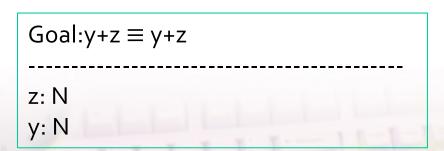
加法的结合律

```
+assoc : \forall (x y z : \mathbb{N}) \rightarrow x + (y + z) \equiv (x + y) + z
+assoc zero y z = refl
+assoc (suc x) y z rewrite +assoc x y z = refl
```

如何通过"hole"来交互式地开发以上的证明?

步骤2: Ctrl+c Ctrl+, 观察正规化后的goal和 context

+assoc zero y
$$z = \{! \ 0!\}$$





Associativity and Working with Holes

加法的结合律

```
+assoc : \forall (x y z : \mathbb{N}) \rightarrow x + (y + z) \equiv (x + y) + z
+assoc zero y z = refl
+assoc (suc x) y z rewrite +assoc x y z = refl
```

如何通过"hole"来交互式地开发以上的证明?

步骤3: 输入解决方法, Ctrl+c Ctrl+r进行检查 (有时可自动推导出解决方法)

+assoc zero y
$$z = \{! \ 0!\}$$



+assoc zero y z = refl

Exercise: Prove the second case for +assoc.



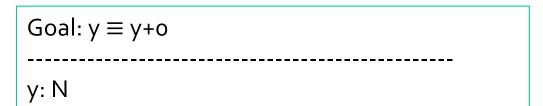
```
+comm : \forall (x y : N) \rightarrow x + y \equiv y + x
+comm zero y = ?
+comm (suc x) y = ?
```

Load file (Ctrl+c Ctrl+l)



+comm : \forall (x y : N) \rightarrow x + y \equiv y + x +comm zero y = { }o

观察hole (Ctrl+c Ctrl+,)



因此,

+comm : \forall (x y : N) \rightarrow x + y \equiv y + x +comm zero y rewrite +o y = refl



```
+comm : \forall (x y : N) \rightarrow x + y \equiv y + x
+comm zero y rewrite +o y = refl
+comm (suc x) y = { }o
```

观察hole (Ctrl+c Ctrl+,)

```
Goal: suc (x + y) \equiv y + suc x

y : N

x : N
```

我们可以利用归纳假设 x+y≡y+x 来重写上面的goal



```
+comm : \forall (x y : N) \rightarrow x + y \equiv y + x
+comm zero y rewrite +o y = refl
+comm (suc x) y rewrite +comm x y = { }o
```

观察hole (Ctrl+c Ctrl+,)

```
Goal: suc (y + x) \equiv y + suc x

y : N

x : N
```

证明辅助引理: suc (y + x) ≡ y + suc x?

```
+suc : \forall (x y : N) \rightarrow x + (suc y) \equiv suc (x + y)
+suc zero y = refl
+suc (suc x) y rewrite +suc x y = refl
```

```
+comm : \forall (x y : N) \rightarrow x + y \equiv y + x
+comm zero y rewrite +o y = refl
+comm (suc x) y rewrite +comm x y = { }o
```

使用辅助引理

```
+comm : \forall (x y : N) \rightarrow x + y \equiv y + x
+comm zero y rewrite +o y = refl
+comm (suc x) y rewrite +suc y x | +comm x y = refl
```



Distributivity of Multiplication and Choosing the Induction Variable

```
*distribr : \forall (x y z : N) \rightarrow (x + y) * z \equiv x * z + y * z 
*distribr x y z = { }o
```

选择递归变量Ctrl+cCtrl+c



```
*distribr : \forall (x y z : N) \rightarrow (x + y) * z \equiv x * z + y * z 
*distribr zero y z = { }0 
*distribr (suc x) y z = { }1
```



Distributivity of Multiplication and Choosing the Induction Variable

```
*distribr : \forall (x y z : N) \rightarrow (x + y) * z \equiv x * z + y * z

*distribr zero y z = { }o

*distribr (suc x) y z = { }1
```

选择递归变量Ctrl+cCtrl+,



$$y * z \equiv y * z$$

Ctrl+c Ctrl+r



```
*distribr : \forall (x y z : N) \rightarrow (x + y) * z \equiv x * z + y * z 
*distribr zero y z = refl
*distribr (suc x) y z = { }1
```



Distributivity of Multiplication and Choosing the Induction Variable

```
*distribr : \forall (x y z : N) \rightarrow (x + y) * z \equiv x * z + y * z
*distribr zero y z = refl
*distribr (suc x) y z = { }o
```

观察goal和context Ctrl+c Ctrl+,

$$z + (x + y) * z \equiv z + x * z + y * z$$

$$p: z + ((x + y) * z) \equiv (z + x * z) + y * z$$

*distribr : \forall (x y z : N) \rightarrow (x + y) * z \equiv x * z + y * z *distribr zero y z = refl *distribr (suc x) y z rewrite *distribr x y z = +assoc z (x * z) (y * z)



Arithmetic Comparison

```
0 < 0 = ff
0 < (suc y) = tt
(suc x) < (suc y) = x < y
(suc x) < 0 = ff
\underline{\hspace{1cm}} = \mathbb{N} \underline{\hspace{1cm}} : \hspace{1cm} \mathbb{N} \to \mathbb{N} \to \mathbb{B}
0 =N 0 = tt
suc x = \mathbb{N} suc y = x = \mathbb{N} y
_{-} =\mathbb{N} _{-} = ff
\underline{\leq} \mathbb{N} \to \mathbb{N} \to \mathbb{B}
x \le y = (x < y) \mid |x = N y
_{} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{B}
a > b = b < a
\ge \mathbb{N} \to \mathbb{N} \to \mathbb{B}
a \ge b = b \le a
```

Arithmetic Comparison

自然数不可能小于o

```
<-0: \forall (x: \mathbb{N}) \rightarrow x < 0 \equiv ff
<-0: 0 = refl
<-0: (suc y) = refl
```

传递性

```
<-trans : \forall \{x \ y \ z : \mathbb{N}\} \rightarrow x < y \equiv tt \rightarrow y < z \equiv tt \rightarrow x < z \equiv tt
<-trans \{x\} \ \{0\} \ p1 \ p2 \ rewrite < -0 \ x = \mathbb{B}-contra \ p1
<-trans \{0\} \ \{suc \ y\} \ \{0\} \ p1 \ ()
<-trans \{0\} \ \{suc \ y\} \ \{suc \ z\} \ p1 \ p2 = refl
<-trans \{suc \ x\} \ \{suc \ y\} \ \{0\} \ p1 \ ()
<-trans \{suc \ x\} \ \{suc \ y\} \ \{suc \ z\} \ p1 \ p2 = <-trans \{x\} \ \{y\} \ \{z\} \ p1 \ p2
```

```
\mathbb{B}-contra : ff ≡ tt → \forall \{\ell\} {P : Set \ell\} → P \mathbb{B}-contra ()
```



Dotted Variables

```
<-trans : \forall \{x \ y \ z : N\} \rightarrow
x < y \equiv tt \rightarrow y < z \equiv tt \rightarrow x < z \equiv tt
<-trans p q = \{\} o
```

观察goal和context Ctrl+c Ctrl+,

```
Goal: .x < .z \equiv tt

p: .x < .y \equiv tt

q: .y < .z \equiv tt

.z: N

.y: N

.x: N
```



Dotted Variables

```
<-trans: \forall \{x \ y \ z : N\} \rightarrow
 x < y \equiv tt \rightarrow y < z \equiv tt \rightarrow x < z \equiv tt
<-trans{x}{y}{z} p q = { }0
```

观察goal和context Ctrl+c Ctrl+,

```
Goal: x < z \equiv tt

p: x < y \equiv tt

q: y < z \equiv tt

z: N

y: N

x: N
```



An Equality Test

A function f of type $A \rightarrow A \rightarrow B$ is defined to be an equality test when f x y returns tt if and only if $x \equiv y$ is provable.

如何证明_=N_ 是一个相等测试。



An Equality Test

A function f of type $A \rightarrow A \rightarrow B$ is defined to be an equality test when f x y returns tt if and only if $x \equiv y$ is provable.

```
= \mathbb{N} - to - \equiv : \forall \{x \ y : \mathbb{N}\} \rightarrow x = \mathbb{N} \ y \equiv tt \rightarrow x \equiv y
= \mathbb{N} - to - \equiv \{0\} \ \{0\} \ u = refl
= \mathbb{N} - to - \equiv \{suc \ x\} \ \{0\} \ ()
= \mathbb{N} - to - \equiv \{suc \ x\} \ \{suc \ y\} \ u \ rewrite = \mathbb{N} - to - \equiv \{x\} \ \{y\} \ u = refl
```

```
=N-from-\equiv : \forall {x y : \mathbb{N}} \rightarrow x \equiv y \rightarrow x =\mathbb{N} y \equiv tt =N-refl : \forall (x : \mathbb{N}) \rightarrow (x =\mathbb{N} x) \equiv tt =N-refl 0 = refl =N-refl x
```



Mutually Recursive Definitions

相互递归的函数定义

```
is-even : N → B
is-odd : N → B
is-even 0 = tt
is-even (suc x) = is-odd x
is-odd 0 = ff
is-odd (suc x) = is-even x
```

相互递归的证明

```
even~odd : \forall (x : \mathbb{N}) \rightarrow is-even x \equiv \sim is-odd x odd~even : \forall (x : \mathbb{N}) \rightarrow is-odd x \equiv \sim is-even x even~odd zero = refl even~odd (suc x) = odd~even x odd~even zero = refl odd~even (suc x) = even~odd x
```



Homework

18.1. 证明 _*_ 满足交换性和结合律。 ∀ {x y : N} → x * y ≡ y * x ∀ {x y z : N} → x * (y * z) ≡ (x * y) * z

18.2. 证明下面的性质。

```
\forall (n : \mathbb{N}) \rightarrow n < n \equiv ff
\forall \{x y : \mathbb{N}\} \rightarrow x < y \equiv tt \rightarrow y < x \equiv ff
\forall (n m : \mathbb{N}) \rightarrow n < m \mid \mid n = \mathbb{N} m \mid \mid m < n \equiv tt
```

