INDUCTION

FRANK TSAI

Contents

1. Induction

Roughly speaking, natural numbers $\mathbb{N}$ are the "minimal" data type equipped with a successor function $s : \mathbb{N} \to \mathbb{N}$. We can define $\mathbb{N}$ inductively:

(i) 0 is a natural number;
(ii) if n is a natural number, then $s(n)$ is also a natural number.

```
data ℕ : Set where
  zero : ℕ
  suc : ℕ → ℕ
```

We may want to prove that every natural number has some property P. For example, we may want to prove that for any natural number n, $2 \mid n(n+1)$. If we proceed directly, we may write down "let n be a natural number, we need to prove $2 \mid n(n+1)$." Sadly, we are stuck because we don't know what n is.

The good news is we know that every natural number is either 0 or the successor of some other natural number, i.e., $s(m)$. If we can somehow show the followings:

$$P(0) \qquad\qquad \forall k.\,(P(k) \Rightarrow P(s(k)))$$

then we can reason as follows: let n be any natural number, we want to show $P(n)$. We can do a case analysis on n. If n is 0, then we need to show $P(0)$, which we have shown already. If n is $s(m)$ instead, then we need to show $P(s(m))$. Since we know $\forall k.\,(P(k) \Rightarrow P(s(k)))$, it suffices to show $P(m)$. To show $P(m)$, we repeat the previous argument, i.e., we do a case analysis on m and so on.

This is similar to writing a recursive function with a base case in programming. In fact, we can implement this in programming languages such as Agda.

```
induction : {P : ℕ → Set} →
            P 0 →
            (∀ k → P k → P (suc k)) →
            ∀ n → P n
```

Date: November 1, 2023.

$\mathsf{induction}\ b\ f\ \mathsf{zero} = b$
$\mathsf{induction}\ b\ f\ (\mathsf{suc}\ x) = f\ x\ (\mathsf{induction}\ b\ f\ x)$

In summary, the principle of induction says that to prove that $\forall n. P(n)$, it suffices to prove two things:

(i) Base case: $P(0)$, and
(ii) Induction step: $\forall k. (P(k) \Rightarrow P(s(k)))$.

To prove the induction step, we introduce a natural number k and assume $P(k)$ (the induction hypothesis), and derive $P(s(k))$.

2. Examples

Proposition 2.1. *For all $n \in \mathbb{N}$, $2 \mid n(n+1)$.*

Proof. $P(n)$ is $2 \mid n(n+1)$. By induction on n, it suffices to prove

(i) Base case: $P(0)$, i.e., $2 \mid 0(0+1)$.
(ii) Induction step: $\forall k. (P(k) \Rightarrow P(k+1))$, i.e.,

$$\forall k. (2 \mid k(k+1) \Rightarrow 2 \mid (k+1)((k+1)+1))$$

(Base case): Clearly, 2 divides 0.

(Induction step): Let k be a natural number. Assume $2 \mid k(k+1)$. We need to show that $2 \mid (k+1)((k+1)+1)$, or equivalently, $2 \mid (k(k+1)+2k+2)$. By the induction hypothesis, 2 divides $k(k+1)$, and clearly, 2 also divides $2k+2$. $\square$

Proposition 2.2. *For all $n \in \mathbb{N}$, $2^0 + 2^1 + \cdots + 2^n = 2^{n+1} - 1$, i.e.,*

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

Proof. $P(n)$ is $2^0 + 2^1 + \cdots + 2^n = 2^{n+1} - 1$. By induction on n, it suffices to prove

(i) Base case: $P(0)$, i.e., $2^0 = 2^{0+1} - 1$.
(ii) Induction step: $\forall k. (P(k) \Rightarrow P(k+1))$, i.e.,

$$\forall k. \left(\sum_{i=0}^{n} 2^i = 2^{n+1} - 1 \Rightarrow \sum_{i=0}^{n+1} 2^i = 2^{(n+1)+1} - 1 \right)$$

(Base case): It follows immediately by computation.

(Induction case): Let k be a natural number. Assume that

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

We need to prove

$$\sum_{i=0}^{n+1} 2^i = 2^{(n+1)+1} - 1$$

Using the induction hypothesis, the left hand side can be rewritten as follows:

$$\left(\sum_{i=0}^{n} 2^i \right) + 2^{n+1} = 2^{n+1} - 1 + 2^{n+1} = 2^{n+2} - 1$$

$\square$

So far, the choice of P is straightforward. Let's see an example where the choice of P is not so straightforward, but let's see what happens if we choose the naïve P.

Proposition 2.3. *Consider a function $f : \mathbb{N} \to \mathbb{N}$ defined recursively as follows:*

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 3 \\ f(n+2) &= 2f(n+1) - f(n) \end{aligned}$$

This function has a closed form:

$$\forall n.\, f(n) = 2n + 1$$

Failed Attempt. $P(n)$ is $f(n) = 2n + 1$. By induction on n, we need to prove the base case and the induction step.

(Base case): $P(0)$ is $f(0) = 2 \cdot 0 + 1$. By definition, $f(0) = 1$ and by computation $2 \cdot 0 + 1 = 1$, so the base case goes through fine.

(Induction step): Let $k \in \mathbb{N}$. Assume that $f(k) = 2k + 1$, we need to prove $f(k+1) = 2(k+1) + 1$. If k is 0, then the equality follows by computation. If $k > 0$, then by definition $f(k+1) = 2f(k) - f(k-1)$. By the induction hypothesis, $f(k) = 2k + 1$, so

$$f(k+1) = 2(2k+1) - f(k-1)$$

We are stuck because the induction hypothesis does not tell us anything about $f(k-1)$. □

We need a stronger induction hypothesis that tells us something about $f(k-1)$. This requires a different choice of P. Let's consider the following lemma.

Lemma 2.4. *For all $m \in \mathbb{N}$ and $n \in \mathbb{N}$, if $n < m$ then $f(n) = 2n + 1$.*

Proof. $P(m)$ is $\forall n.\, (n < m \Rightarrow f(n) = 2n + 1)$. By induction on m, it suffices to show the base case and the induction step.

(Base case): $P(0)$ is $\forall n.\, (n < 0 \Rightarrow f(n) = 2n + 1)$. Let $n \in \mathbb{N}$. Assume $n < 0$. This is a contradiction because no natural number is strictly less than 0.

(Induction step): Let $k \in \mathbb{N}$. The induction hypothesis $P(k)$ is

$$\forall n.\, (n < k \Rightarrow f(n) = 2n + 1)$$

and we need to prove $P(k+1)$, which is

$$\forall n.\, (n < k + 1 \Rightarrow f(n) = 2n + 1)$$

Note that the induction hypothesis now tells us something about $f(n)$ for any n less than k. To prove $P(k+1)$, let $n \in \mathbb{N}$. Assume $n < k + 1$.

If k is 0 or 1, then n is 0 or 1. These two cases follow directly from how f is defined.

For $k > 1$, there are two cases: If $n < k$, then the result follows immediately from the induction hypothesis.

If $n = k$, then by definition $f(k) = 2f(k-1) - f(k-2)$. Since $k - 2 < k$ and $k - 1 < k$, the induction hypothesis says that $f(k-2) = 2(k-2) + 1 = 2k - 3$ and that $f(k-1) = 2(k-1) + 1 = 2k - 1$. Thus,

$$\begin{aligned} f(k) &= 2f(k-1) - f(k-2) \\ &= 2(2k-1) - (2k-3) \\ &= 4k - 2 - 2k + 3 \\ &= 2k + 1 \end{aligned}$$

□

Proposition 2.3 is an immediate corollary of Lemma 2.4.

Proof of Proposition 2.3. Let $n \in \mathbb{N}$. We need to prove $f(n) = 2n+1$. This follows immediately from Lemma 2.4 by setting $m := n + 1$. $\square$

You may have heard *strong induction* in class. Unfortunately, the name "strong induction" is somewhat misleading because anything provable with strong induction can be proved with mathematical induction presented here and vice-versa, i.e., strong induction is **not** stronger than mathematical induction. In fact, the pattern used in Lemma 2.4 is what strong induction does.

Proposition 2.5. *For any $n \in \mathbb{N}$, if $n \geq 2$ then n is a linear combination of 2 and 3, i.e., there are natural numbers i and j so that $n = 2i + 3j$.*

Again, we need to strengthen the induction hypothesis.

Lemma 2.6. *For any $m \in \mathbb{N}$ and $n \in \mathbb{N}$, if $n < m$ then if additionally $n \geq 2$ then n is a linear combination of 2 and 3.*

Proof. $P(m)$ is $\forall n. (n < m \Rightarrow (n \geq 2 \Rightarrow \exists i. \exists j. n = 2i + 3j))$.

(Base case): $P(0)$ is $\forall n. (n < 0 \Rightarrow (n \geq 2 \Rightarrow \exists i. \exists j. n = 2i + 3j))$. Let $n \in \mathbb{N}$. Assume $n < 0$. This is a contradiction.

(Induction step): Let $k \in \mathbb{N}$. Assume $P(k)$, i.e.,

$$\forall n. (n < k \Rightarrow (n \geq 2 \Rightarrow \exists i. \exists j. n = 2i + 3j))$$

We need to prove $P(k+1)$, i.e.,

$$\forall n. (n < k + 1 \Rightarrow (n \geq 2 \Rightarrow \exists i. \exists j. n = 2i + 3j))$$

Let $n \in \mathbb{N}$. Assume $n < k + 1$ and $n \geq 2$, so $k \geq 2$. If $k = 2$ then n has to be 2, which can be expressed as $2 \cdot 1 + 3 \cdot 0$. If $k = 3$ then n has to be 2 or 3. We know how to express 2, and 3 can be expressed as $2 \cdot 0 + 3 \cdot 1$. For $k \geq 4$, if $n < k$ then the induction hypothesis gives us what we want. If $n = k$, then consider $k - 2$. The induction hypothesis tells us that there are natural numbers a and b so that $k - 2 = 2a + 3b$. We can then express n as $n = k = k - 2 + 2 = 2(a + 1) + 3b$. $\square$

Proof of Proposition 2.5. Let $n \in \mathbb{N}$. Assume $n \geq 2$. The result follows immediately from Lemma 2.6. $\square$

3. Well-Founded Induction

We can generalize this argument to any data type equipped with a *well-founded* relation. A binary relation R on S is well-founded if every element s of S is *accessible*. An element s is said to be accessible if it does not have an infinite descending chain with respect to R, i.e., s reaches a base case in finitely many steps.

```
data acc {A : Set} (r : A → A → Set) : A → Set where
  acc_k : (x : A) → ((y : A) → r y x → acc r y) → acc r x

wf : {A : Set} → (A → A → Set) → Set
wf {A} r = (x : A) → acc r x
```

Well-founded induction says that to prove $\forall x. P(x)$, it suffices to prove

$$\forall x. ((\forall y. yRx \to P(y)) \to P(x))$$

where R is a well-founded relation.

```
wf-induction : {A : Set} →
               {P : A → Set} →
               (_<_ : A → A → Set) →
               wf _<_ →
               ((x : A) → ((y : A) → y < x → P y) → P x) →
               (x : A) → P x
wf-induction {A} {P} _<_ wfp f x = h x (wfp x) where
  h : (x : A) → acc _<_ x → P x
  h x (acc .x k g) = f x (λ y l → h y (g y l))
```

In particular, strong induction is a special case of well-founded induction because the less-than relation on ℕ is well-founded.

```
data _<_ : Nat → Nat → Set where
  zero_suc : (n : Nat) → 0 < suc n
  n_suc : (n m : Nat) → n < m → n < suc m

private
  <-0-acc : acc _<_ 0
  <-0-acc = acc 0 k (λ _ → h) where
    h : {y : Nat} → y < 0 → acc _<_ y
    h ()

  <-1-acc : acc _<_ 1
  <-1-acc = acc 1 k (λ _ → h) where
    h : {y : Nat} → y < 1 → acc _<_ y
    h (zero_suc .0) = <-0-acc

  <-suc-acc : (x : Nat) → acc _<_ x → acc _<_ (suc x)
  <-suc-acc zero _ = <-1-acc
  <-suc-acc (suc x) (acc .(suc x) k e) = acc (suc (suc x)) k h where
    h : (y : Nat) → y < suc (suc x) → acc _<_ y
    h zero l = <-0-acc
    h (suc y) (n .(suc y) suc .(suc x) l) = e (suc y) l

<-wf : wf _<_
<-wf zero = <-0-acc
<-wf (suc x) = <-suc-acc x (<-wf x)

strong-induction : {P : Nat → Set} →
                   ((x : Nat) → ((y : Nat) → y < x → P y) → P x) →
                   (x : Nat) → P x
strong-induction = wf-induction _<_ <-wf
```

In summary, strong induction says that to prove $\forall x \in \mathbb{N}.\, P(x)$, it suffices to prove

$$\forall k.\, ((\forall y.\, y < k \to P(y)) \to P(k))$$