

5 Induction and Recursion

5.1 Mathematical Induction

The First Principle of Mathematical Induction

$P(0)$

basis

$\forall n \geq 0 P(n) \rightarrow P(n+1)$

induction

$\therefore \forall n \geq 0 P(n).$

proof

$P(0).$

$P(0) \rightarrow P(1).$

$P(1) \rightarrow P(2).$

...

Q.E.D.

domino effect

5.2 Strong Induction and Well Ordering

The Second Principle of Mathematical Induction (Strong Induction)

$P(0)$ *basis*
 $\forall n \geq 0: (\forall k: 0 \leq k \leq n P(k)) \rightarrow P(n+1)$ *induction*
 $\therefore \forall n \geq 0 P(n).$

Why strong?

$\forall n \geq 0: (\forall k: 0 \leq k \leq n P(k))$ is **stronger** (\supseteq) than $P(n)$.

Ex. 4, Thm 1 (중학교기하)

Generalized Induction

Let $C \geq 1, c \geq 0$.

$\forall n: c \leq n \leq c+C-1, P(n)$ *C-basis*

$\forall n: n \geq c, P(n) \rightarrow P(n+C)$ *C-step induction*

$\therefore \forall n \geq c P(n).$

Proofs Using Well Ordering Property

well ordering property

Every nonempty set of nonnegative integers has a least element.

$$\forall S: \emptyset \subset S \subseteq \mathbf{N}, \exists m \in S .\exists. \forall n \in S, m \leq n.$$

m: least element

proof Assume $S = \{n \mid \neg P(n)\} \supset \emptyset$ and $m \in S$, is the least element.

*m ≠ 0, since P(0) by **basis**.*

m - 1 ∈ N, but m - 1 ∉ S.

*P(m - 1) → P(m) is **contradicted by induction**.*

Exa. 5 Let $a \in \mathbf{Z}$, $d \in \mathbf{N}^+$. Then

$\exists^1 q$ (quotient), r (remainder) $\in \mathbf{Z}$, $\exists. n = dq + r, 0 \leq r < d$.

proof Let $X = \{n-dq \mid n-dq \geq 0, q \in \mathbf{Z}\}$

$X \neq \emptyset$, since you can make negative integer q with large absolute value.

$\therefore \exists$ the **least element** $r = n-dq_0 \in X$.

5.3 Recursive Definition and Structural Induction

Recursively Defined Function

$f: \mathbf{N} \rightarrow S$ (for any set S)

i) Define $f(0)$.

ii) For $n > 0$, define $f(n)$ in terms of $f(0), f(0), \dots, f(n-1)$.

Example Define the series $a_n := a^n$ recursively:

Let $a_0 := 1$ $a^0 = 1$

For $n > 0$, $a_n := a \cdot a_{n-1}$. $a^n = a \cdot a^{n-1}$ for $n > 0$

Example 2 $F(n) = n!$

$F(0) = 1$. $0! = 1$

$F(n) = n \cdot F(n-1)$ for $n > 0$. $n! = n \cdot (n-1)!$ for $n > 0$

Definition 1 The Fibonacci series are defined by $f_0 = 0$, $f_1 = 1$, and

$$f_n = f_{n-1} + f_{n-2} \quad \text{for } n \geq 2.$$

Theorem $f_n < 2^n$.

Proof Mathematical induction

basis $f_0 = 0 < 2^0 = 1$ and $f_1 = 1 < 2^1 = 2$.

induction Assume $\forall k < n, f_k < 2^k$. (strong induction)

$$f_n = f_{n-1} + f_{n-2} < 2^{n-1} + 2^{n-2} < 2^{n-1} + 2^{n-1} = 2^n.$$

Example

Recursively Defined Sets and Structures

Definition 2 The set of Σ^* of strings over the alphabet Σ is defined:

basis $\varepsilon \in \Sigma^*$ where ε is the empty string containing no symbols.

recursion If $w \in \Sigma^*$ and $a \in \Sigma$, then $wa \in \Sigma^*$.

Example 9 The length of a string $|w|$, $w \in \Sigma^*$.

basis $|\varepsilon| = 0$.

recursion If $w \in \Sigma^*$ and $a \in \Sigma$, then $|wa| = |w| + 1$.

Definition Set of natural numbers \mathbf{N} .

basis $0 \in \mathbf{N}$.

recursion If $n \in \mathbf{N}$, then $n+1 \in \mathbf{N}$.

Example 10 Well-Formed Formulae for Compound Statement Form

basis \mathbf{T} , \mathbf{F} , and s where s propositional variables are w.f.f..

recursion If E and F are well-formed formulae, $\neg E$, $E \wedge F$, $E \vee F$, $E \rightarrow F$, $E \leftrightarrow F$, and (E) are well-formed formulae.

Abstract syntax

$$E ::= \mathbf{T} \mid \mathbf{F} \mid s \mid \neg E \mid E \wedge E \mid E \vee E \mid E \rightarrow E \mid E \leftrightarrow E \mid (E)$$

Example 11 Well-Formed Formulae for Expression

$$E ::= n \mid v \mid E + E \mid E - E \mid E * E \mid E / E \mid E \uparrow E \mid (E)$$

where n is integer, v is a variable, and \uparrow denotes exponential.

Definition 4 The set of **rooted trees** with root r .

basis A single vertex r is a rooted tree with root r .

recursion Suppose T_1, T_2, \dots, T_n are disjoint rooted trees with roots r_1, r_2, \dots, r_n , respectively. Adding a **new** node r and edges to each of the vertices r_1, r_2, \dots, r_n is also rooted tree with root r .

A **directed graph (digraph)** $G = (V, E)$

where V is a set of **vertices** and $E \subseteq V \times V$ is a set of **edges**.

The rooted tree $T = (V, E, r)$ is a digraph (V, E) with root $r \in V$, and

Definition 4.5 The set of **rooted trees** with root r is defined as follows:

basis $(\{r\}, \emptyset, r)$ is a rooted tree.

recursion If $(V_1, E_1, r_1), (V_2, E_2, r_2), \dots, (V_n, E_n, r_n)$ are rooted trees

where $1 \leq i < j \leq n: V_i \cap V_j = \emptyset$. Then **disjoint**

$(\cup_{i=1}^n V_i \cup \{r\}, \cup_{i=1}^n E_i \cup \{(r, r_1), (r, r_2), \dots, (r, r_n)\}, r)$ is a rooted tree.

Definition 5 The set of **extended binary trees (EBT)** with root r .

basis An empty set is an extended binary tree.

recursion If T_1 and T_2 are disjoint EBT. Adding a new node r and edges to each of the roots of left subtree T_1 and right subtree T_2 is an EBT with root r when these subtrees are not empty.

Definition 6 The set of **full binary trees (FBT)** with root r .

basis A full binary tree with root r .

recursion If T_1 and T_2 are disjoint FBT. Adding a new node r and edges to each of the roots of left subtree T_1 and right subtree T_2 is an FBT with root r .

Definition 6 The set of **full binary tree (FBT)** with root r is defined as ...

basis $(\{r\}, \emptyset, r)$ is a FBT.

recursion If $T_1 = (V_1, E_1, r_1)$ and $T_2 = (V_2, E_2, r_2)$ are FBT's where ...

$T = T_1 \cdot T_2 = (V_1 \cup V_2 \cup \{r\}, E_1 \cup E_2 \cup \{(r, r_1), (r, r_2)\}, r)$ is a FBT.

Structural Induction

Example 12 Let $3 \in S$, and let $x+y \in S$, if $x, y \in S$. Let $A = \{n \in \mathbf{Z}^+ \mid (3|n)\}$

Prove $A = S$

i) Prove $A \subseteq S$.

base: Let $n = 3 \in A$, $3 \in S$ by basis def'n of S .

induction: if $n \in A$, then $n \in S$.

Consider $n + 3$. $n+3 \in S$, since $3 \in S$.

ii) Prove $S \subseteq A$,

base: $3 \in S$, $3|3$, $3 \in A$.

recursion: Consider $n = x+y \in S$.

*$\forall x < n, \forall y < n: x, y \in A$. *strong induction hypothesis**

$\therefore 3 \mid x \wedge 3 \mid y$. $3 \mid (x+y)$.

$x+y \in A$.

Definition 7 Height of a full balanced tree.

basis $h(\{r\}, \emptyset, r) = 0$.

recursion $h(T_1 \cdot T_2) = 1 + \max(h(T_1), h(T_2))$.

Definition Number of nodes of a full balanced tree.

basis $n(\{r\}, \emptyset, r) = 1$.

recursion $n(T_1 \cdot T_2) = 1 + n(T_1) + n(T_2)$.

Theorem 2 Let T be a full binary tree. Then $n(T) \leq 2^{h(T)+1} - 1$.

basis step $h(\{r\}, \emptyset, r) = 0, n(\{r\}, \emptyset, r) = 1. \therefore 1 \leq 2^1 - 1 = 1$.

recursion step Assume $n(T_1) \leq 2^{h(T_1)+1} - 1, n(T_2) \leq 2^{h(T_2)+1} - 1$.

$$\begin{aligned} n(T) &= 1 + n(T_1) + n(T_2) && \leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) \\ &&& \leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 && = 2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1 \\ &&& = 2^{h(T)+1} - 1 \end{aligned}$$

Lexicographic ordering(Example 15)

Let $< \subseteq \mathbf{N} \times \mathbf{N}$. $(x_1, y_1) < (x_2, y_2)$, if $(x_1 < x_2) \vee ((x_1 = x_2) \wedge (y_1 < y_2))$.

Every subset of $\mathbf{N} \times \mathbf{N}$ has a **least element(well-ordering)**

Define $a_{m,n}: \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$ **recursively.**

basis $a_{0,0} = 0$.

recursion $a_{m,n} = a_{m-1,n} + 1$ if $(n = 0) \wedge (m > 0)$

$a_{m,n-1} + n$ if $(n > 0)$

	0	1	2	3	4	\dots	m	$m+1$
0	0	1	3	6	10	\dots	$K+m$	$K+m+1$
1	1	4	6	9	\dots		$K+m-1$	$K+1+m$
2	2	3	5	\dots		$K+m-2$	$K+2+m-1$	
3	3	4	\dots		$K+m-3$			
\dots	\dots							
m	m	$K+m+1$						
$m+1$	$m+1$							

Solution We can prove $a_{m,n} = m + n(n+1)/2$, if $a_{i,j} < a_{m,n}$, $(i, j) < (m, n)$.

basis $a_{0,0} = 0 + 0 \cdot 1/2 = 0$. $a_{0,0}$ and $(0, 0)$ is a least element in \mathbf{N} and $\mathbf{N} \times \mathbf{N}$.

recursion Suppose $(0, 0) < \forall (i, j) < (m, n): a_{i,j} = i + j(j+1)/2$

case 1) If $(n = 0) \wedge (m > 0)$ $(m-1, 0) < (m, 0)$ (I.H.)

$$a_{m-1,0} = m - 1 + 0(0+1)/2 = m - 1 \quad \text{Induction hypothesis}$$

$$a_{m,0} = a_{m-1,0} + 1 = m - 1 + 1 = m. \quad \text{Recursive def. of } a_{m,0}.$$

$$a_{m,0} = m + 0(0+1)/2 = m.$$

case 2) If $(n > 0)$ $(m, n-1) < (m, n)$

$$a_{m,n-1} = m + n(n-1)/2 \quad a_{m,n} = m + n(n-1)/2 + n = m + n(n+1)/2.$$

5.4 Recursive Algorithms

Recursive functions

Recursive sets

Recursive algorithm

Algorithm 1 Recursive Algorithm for Computing $n!$

function $factorial(n: \mathbf{N}): \mathbf{N};$

if $n=0 \rightarrow factorial(n) := 1$

 | $n \geq 1 \rightarrow factorial(n) := n \cdot factorial(n-1)$

fi

Algorithm 2 Recursive Algorithm for Computing a^n

function $power(a: \mathbf{Z}^+, n: \mathbf{N};): \mathbf{N};$

if $n=0 \rightarrow power(a, n) := 1$

 | $n \geq 1 \rightarrow power(a, n) := a \cdot power(a, n-1)$

fi

Compute $b^n \bmod m$ where $m \geq 2$, $n \geq 0$, $1 \leq b < m$.

$$b^n \bmod m = (b \cdot (b^{n-1} \bmod m)) \bmod m.$$

$$b^0 \bmod m = 1.$$

$$O(n)$$

But

$$b^n \bmod m = (b^{n/2} \bmod m)^2 \bmod m.$$

Algorithm 3 Recursive Modular Exponentiation

function $mpower(b, n, m: \mathbf{N}, n: \mathbf{N};): \mathbf{N};$

if $n=0 \rightarrow mpower(b, n, m) := 1$

| $(2 \mid n) \rightarrow mpower(b, n, m) := mpower(b, n/2, m)^2 \bmod m$

| $(2 \nmid n) \rightarrow mpower(b, n, m) :=$

$(mpower(b, \lfloor n/2 \rfloor, m)^2 \bmod m \cdot b \bmod m) \bmod m$

fi

function (recursive) fibonacci($n \in \mathbf{N}$) $\in \mathbf{N}$;

if $n = 0 \rightarrow \text{fibonacci} := 0$;

 | $n = 1 \rightarrow \text{fibonacci} := 1$;

 | $n \geq 2 \rightarrow \text{fibonacci}(n-1) + \text{fibonacci}(n-2)$

fi

Theorem *fibonacci*(n) performs $f_{n+1} - 1$ additions

basis *fibonacci*(0): $f_{0+1} - 1 = f_1 - 1 = 1 - 1 = 0$ additions.

fibonacci(1): $f_{1+1} - 1 = f_2 - 1 = 1 - 1 = 0$ additions.

recursion Assume *fibonacci*($n-1$): $f_n - 1$; *fibonacci*($n-2$): $f_{n-1} - 1$ addi.

fibonacci(n): $(f_n - 1) + (f_{n-1} - 1) + 1 = f_n + f_{n-1} - 1 = f_{n+1} - 1$ additions.

function (iterative) fibo($n \in \mathbf{N}$) $\in \mathbf{N}$;

if $n = 0 \rightarrow \text{fibo} := 0$

 | $n = 1 \rightarrow \text{fibo} := 1$

 | $n \geq 2 \rightarrow x, y := 0, 1$;

for $i:=2$ **to** n **do** $\text{fibo}, x, y := x+y, y, \text{fibo}$ **od** **fi**

function $\text{findFib}(a, b, i, n \in \mathbf{N}) \in \mathbf{N}$;
 { Assert : $(a = f_{i-2}) \wedge (b = f_{i-1}) \wedge (2 \leq i \leq n)$ }
 if $i = n \rightarrow \text{findFib} := b$;
 | $i \neq n \rightarrow \text{findFib} := \text{findFib}(b, a+b, i+1, n)$
 fi

function $\text{fastFib}(n \in \mathbf{N}) \in \mathbf{N}$;
 if $n = 0 \rightarrow \text{fastFib} := 0$
 | $n = 1 \rightarrow \text{fastFib} := 1$
 | $n \geq 2 \rightarrow \text{findFib}(0, 1, 2, n)$
 fi

Storage(assignment statement) for speed efficiency!

<i>recursive fibonacci</i>	$O(2^n)$
<i>iterative fibonacci</i>	$O(n)$
<i>fast recursive fibonacci</i>	$O(n)$

The Merge Sort

function $sort(L = l_1, l_2, \dots, l_n \in \mathbf{N}^n) \in \mathbf{N}^n$;

if $n > 1 \rightarrow m := \lfloor n/2 \rfloor$; $L = merge(sort(l_1, \dots, l_m), sort(l_{m+1}, \dots, l_n))$

 | $n \leq 1 \rightarrow sort := L$;

fi

$O(\log n)$

function $merge(A = (a_1, \dots, a_m) \in \mathbf{N}^m; B = (b_1, \dots, b_n) \in \mathbf{N}^n) \in \mathbf{N}^{m+n}$;

 { Assert $(a_1 \leq a_2 \leq \dots \leq a_m) \wedge (b_1 \leq b_2 \leq \dots \leq b_n)$ }

if $m = 0 \rightarrow merge := B$

 | $n = 0 \rightarrow merge := A$

elsif | $a_1 \leq b_1 \rightarrow merge := (a_1, merge((a_2, \dots, a_m); B))$

 | $b_1 \leq a_1 \rightarrow merge := (b_1, merge(A; (b_2, \dots, b_m)))$

fi

$O(n)$

function *iterative merge*($A = a_1, \dots, a_m \in \mathbf{N}^m; B = b_1, \dots, b_n \in \mathbf{N}^n$) $\in \mathbf{N}^{m+n}$;

$L = \text{empty list}; i, j, k, := 1, 1, 1;$

do $(i \leq m) \wedge (j \leq n) \rightarrow$ **if** $i > m \rightarrow L_k := b_j; j, k := j + 1, k + 1$

| $j > n \rightarrow L_k := a_i; i, k := i + 1, k + 1$

| $a_i \leq b_j \rightarrow L_k := a_i; i, k := i + 1, k + 1$

| $a_i \geq b_j \rightarrow L_k := b_j; j, k := j + 1, k + 1$

fi

od;

iterative merge := L .

$O(n \log n)$

do $k \leq m+n \rightarrow$ **if** $(i > m) \vee (a_i \geq b_j) \rightarrow L_k := b_j; j, k := j + 1, k + 1$

| $(j > n) \vee (a_i \leq b_j) \rightarrow L_k := a_i; i, k := i + 1, k + 1$

fi

od

5.5 Program Correctness

Definition 1 A Program S is said to be **partially correct**, with respect to the **initial assertion** p and the **final assertion** q , if the **initial condition** p is true and S **terminates**, then the **final assertion** is true, written $p\{S\}q$ and it is called **Hoare triple**.

Example 1 $(x=1) \{y:=2; z:=x+y\} (z=3)$
 $\mathbf{T} \{y:=2; z:=x+y\} (z=x+2)$

Sequence of statement

$$\begin{array}{l} p \{S_1\} q \\ q \{S_2\} r \\ \hline \therefore p \{S_1; S_2\} r \end{array}$$

If statement

$$\frac{(p \wedge B) \{S\} q \quad (p \wedge \neg B) \rightarrow q}{\therefore p \{ \text{if } B \text{ then } S \} q}$$

If then else statement

$$\frac{(p \wedge B) \{S_1\} q \quad (p \wedge \neg B) \{S_2\} q}{\therefore p \{ \text{if } B \text{ then } S_1 \text{ else } S_2 \} q}$$

while statement

$$\frac{(p \wedge B) \{S\} p}{\therefore p \{\mathbf{while} \ B \ \mathbf{do} \ S\} (p \wedge \neg B)}$$

loop invariance

initialize p

update p

terminate $(p \wedge \neg B)$

procedure multiply(m, n : integer)

$m, n \in \mathbf{Z}$

if $n < 0$ **then** $a := -n$ **else** $a := n$;

$a = |n|$

$k := 0$; $x := 0$;

$x = mk$

do $k < a \rightarrow$

$x := x+n$; $k++$

$x = mk$

od

$x = mk \wedge k=a \therefore x = ma = m|n|$

if $n < 0$ **then** $prod := -x$ **else** $prod := x$; $prod = mn$