5 Induction and Recursion

5.1 Mathematical Induction

The First Principle of Mathematical Induction

$P(1)$ **basis**

$\forall k \geq 1\ P(k) \rightarrow P(k+1)$ **induction**

$\therefore\ \forall n \geq 1\ P(n).$

$(P(1) \wedge \forall k: P(k) \rightarrow P(k+1)) \rightarrow \forall n: P(n) \forall k.$

proof

$P(1).$

$P(2) \rightarrow P(3).$

$P(3) \rightarrow P(4).$

…

Q.E.D.

domino effect

***Ex**. 1, ..., 14.*

Definition** Set of **natural numbers** $\mathbf{N}$. **(Peano Axiom)

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

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

5.2 Strong Induction and Well Ordering

Strong Induction

$P(1)$ **basis**

$[P(1) \wedge P(2) \wedge \ \ldots P(k)) \rightarrow P(k+1)]$ **induction**

$(1 \leq^{\forall} j \leq k\ P(j)) \rightarrow P(k+1)$ **induction**

$\therefore\ \forall n \geq 1\ P(n).$

*Why **strong**?*

$1 \leq^{\forall} j \leq k\ P(j))$ *is **stronger**($\subseteq$) than $P(k)$.*

Thm. 1 A simple polygon with n sides($n \geq 3$),
can be triangulate into (n-2) trialgles.

Generalized Induction

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

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

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

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

$(P(c) \wedge P(c+1) \wedge \ldots P(c+C-1)) \wedge (P(n) \rightarrow P(k+C))$

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

Proofs Using Well Ordering Property

well ordering property

Every ***nonempty*** *set of* ***nonnegative integers*** *has a* ***least*** *element.*

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

m: least element

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

$m \neq 0$, since $P(0)$ by ***basis****.*

$m - 1 \in \mathbf{N}$, but $m - 1 \notin S$.

$P(m-1) \rightarrow P(m)$ is ***contradicted*** *by* ***induction****.*

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

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

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

$X \neq \varnothing$, 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

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

i) Define $f(0)$.

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

Example Define the series $a_n :\equiv a^n$ recursively:

Let $a_0 :\equiv 1$ $\quad a^0 = 1$

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

Example 2 $F(n) = n!$

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

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

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

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

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

$f_0 = 0, f_1 = 1$.

$f_2 = 0+1 = 2$.

$f_3 = 1+2 = 3$.

$f_4 = 2+3 = 5$.

Recursively Defined Sets and Structures

Def. *1 The set of* Σ^* *of* ***strings*** *over the alphabet* Σ*is defined:*

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

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

Def. *2 Let* Σ^* *be set of* ***strings*** *over the alphabet* Σ.

concatenation *of two strings* $\cdot: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$.

basis *If* $w \in \Sigma^*$, $w \cdot \lambda = w$.

rec. *If* $w_1, w_2 \in \Sigma^*$ *and* $x \in \Sigma^*$, *then* $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

Def.*' 2 Let* Σ^* *be set of* ***strings*** *over the alphabet* Σ.

concatenation *of two strings* $\cdot: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$.

basis *If* $w \in \Sigma^*$, $\lambda \cdot w = w$.

rec. *If* $w_1, w_2 \in \Sigma^*$ *and* $x \in \Sigma^*$, *then* $(xw_1) \cdot w_2 = x(w_1 \cdot w_2)$.

Def.*" 2 Let* Σ^* *be set of* ***strings*** *over the alphabet* Σ.

concatenation *of two strings* $\cdot: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$.

basis *If* $w \in \Sigma^*$, $\lambda \cdot w = w$.

rec. *If* $w_1, w_2 \in \Sigma^*$, *then* $(xw_1) \cdot (w_2 y) = x(w_1 \cdot w_2)y$.

Def. *2 Let* Σ^* *be set of* ***strings*** *over the alphabet* Σ.

concatenation *of two strings* $\cdot: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$.

basis *If* $w \in \Sigma^*$, $\lambda \cdot w = w$.

rec. *If* $w_1, w_2 \in \Sigma^*$, *then* $w_1 \cdot w_2 = w_1 \cdot w_2 =_{written} w_1 w_2$

(just tax a posed).

Since $w_1 \cdot (w_2 \cdot w_3) = (w_1 \cdot w_2) \cdot w_3$ ***(assocaiutive)***

$= w_1 \cdot w_2 \cdot w_3 = w_1 w_2 w_3$

Ex. *7 The length of a string* $|w|$, $w \in \Sigma^*$.
basis $|\lambda| = 0$.
recursion *If* $w \in \Sigma^*$ *and* $a \in \Sigma$, *then* $|wa| = |w| + 1$.
Ex. *9' The length of a string* $|w|$, $w, x \in \Sigma^*$.
basis $|\lambda| = 0$.
recursion *If* $w, x \in \Sigma^*$, *then* $|wx| = |w| + |x|$.

$(\Sigma^*, \cdot, \lambda)$ *is a* ***monoid*** ***generated*** *by* Σ.

Four terminolies on string

	element	*set*
single	*symbol*	*alphabet(vocabulary)*
length(n≥0)	*string*	*language*

Ex. *8* ***Well-Formed Formulae*** *in* ***Propositional Logic***
basis **T**, **F**, *and s where s propositional variables are w.f.f..*
recursion *If E and F are well-formed formulae, then* $\neg E$, $E \wedge E$, $E \vee E$, $E \rightarrow E$, $E \leftrightarrow E$, *and (E) are well-formed formulae.*

$\Sigma = \{\mathbf{T}, \mathbf{F}, s, t, \ldots, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\}$
$\Sigma^* = \{\lambda\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$
Is an $x \in \Sigma^*$ *well-formed formula?*
yes or no!
$WFF \subseteq \Sigma^*$
We need a syntax grammar(문법 *) for the language WFF.*

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

Ex.10 ***Well-Formed Formulae*** *for* ***Expressions***

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

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

unambiguous grammar

$E ::= E + T \mid E - T \mid T * F \mid T / F \mid F \uparrow T \mid T$

$T ::= T * F \mid T / F \mid F \uparrow T \mid X$

$F ::= F \uparrow F \mid F$

$X ::= n \mid v \mid (E)$

Def. *3 The set of* ***rooted trees*** *with root r.*

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

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

Def*. The rooted tree T = (V, E, r) is*

(1) V: a set of nodes,

(2)$E \subseteq V \times V$*: is a set of edges(a pair of nodes), and*

(3) $r \in V$ *is a root.*

Def*. 3" The set of* ***rooted trees*** *with root r is defined as follows:*

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

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

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

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

Def. *4 The set of* ***extended binary trees****(EBT) with root r is defined as.*
basis *An empty set is an extended binary tree.*
rec. *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.

Def. *4' The set of* ***extended binary tree****(EBT) with root r is defined as.*
basis *$\varnothing$ is an EBT.*
basis *$(\{r\}, \varnothing, r)$ is an EBT.*
rec. *If $T_1 = (V_1, E_1, r_1)$ and $T_2 = (V_2, E_2, r_2)$ are EBT's where $V_1 \cap V_2 = \varnothing$.*
$T = T_1 \cdot T_2 = (V_1 \cup V_2 \cup \{r\}, E_1 \cup E_2 \cup \{(r, r_1), (r, r_2)\}, r)$ is an EBT.

Def. *5 The set of* ***full binary trees****(FBT) with root r is defined as.*
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.

Def. *5 The set of* ***full binary tree****(FBT) with root r is defined as.*
basis *$(\{r\}, \varnothing, r)$ is a FBT.*
rec. *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 *$h(\{r\}, \varnothing, r) = 0$.*
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 *$n(\{r\}, \varnothing, r) = 1$.*
recursion *$n(T_1 \cdot T_2) = 1 + n(T_1) + n(T_2)$.*

Theorem *2 Let T be a full banary tree. Then $n(T) \le 2^{h(T)+1} - 1$.*
basis step *$h(\{r\}, \varnothing, r) = 1$, $n(\{r\}, \varnothing, r) = 1$. $\therefore 1 \le 2^1 - 1 = 1$.*
recursion step *Assume $n(T_1)) \le 2^{h(T1)+1} - 1$, $n(T_2)) \le 2^{h(T2)+1} - 1$.*

$n(T) = 1 + n(T_1) + n(T_2) \le 1 + (2^{h(T1)+1} - 1) + (2^{h(T2)+1} - 1)$
$\le 2 \cdot \max(2^{h(T1)+1}, 2^{h(T2)+1}) - 1 = 2 \cdot 2^{\max(h(T1), h(T2))+1} - 1$
$= 2^{h(T)+1} - 1$

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	…	m	m+1
0	0	1	3	6	10	…	K+m	K+m+1
1	1	4	6	9	…	K+m-1	K+1+m	
2	2	3	5	…	K+m-2	K+2+m-1		
3	3	4	…	K+m-3				
…	…							
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.)*
$a_{m-1,0} = m - 1 + 0(0+1)/2 = m - 1$ *Induction hypothesis*
$a_{m,0} = a_{m-1,0} + 1 = m - 1 + 1 = m$. *Recursive def. of* $a_{m,0}$.
$a_{m,0} = m + 0(0+1)/2 = m$.
case 2) If $(n > 0)$ $(m, n-1) < (m, n)$
$a_{m,n-1} = m + n(n-1)/2$ $a_{m,n} = m + n(n-1)/2 + n = m + n(n+1)/2$.

5.4 Recursive Algorithms

Recursive functions and sets
Recursive ***algorithm***
Recursive ***theorem***, ***proof***

Alg. *1* ***Recursive*** *Algorithm for Computing* $n!$

function *factorial(n:* **N***):* **N***;*
    if $n=0 \rightarrow$ ***return*** *1*
    $\mid n \geq 1 \rightarrow$ ***return*** $n \cdot factorial(n-1)$
    fi

Alg. *2* ***Recursive*** *Algorithm for Computing* a^n

function *power(a:* $\mathbf{R}^+$*, n:* **N***;):* **N***;*
    if $n=0 \rightarrow$ ***return*** *1*
    $\mid n \geq 1 \rightarrow$ ***return*** $a \cdot power(a, n-1)$
    fi

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

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

b^0 ***mod*** $m = 1$.

$O(n)$

But

b^n ***mod*** $m = (b^{n/2}$ ***mod*** $m)^2$ ***mod*** m.

Alg. *4* ***Recursive*** *Modular Exponention*

function *mpower(b, n, m:* **N***, n:* **N***;):* **N***;*
    if $n=0 \rightarrow$ ***return*** *1*
    $\mid (2 \mid n) \rightarrow$ ***return*** $mpower(b, n/2, m)^2 \bmod m$
    $\mid (2 \nmid n) \rightarrow$ ***return*** $mpower(b, \lfloor n/2 \rfloor, m)^2 \bmod m \cdot b \bmod m) \bmod m$
    fi

$O(\log_2 n)$

Recursion and Iteration

function *(recursive) fibo*$(n \in \mathbf{N}) \in \mathbf{N}$;
  if $n = 0 \rightarrow$ **return** 0;
  | $n = 1 \rightarrow$ **return** 1;
  | $n \geq 2 \rightarrow$ **return** fibo(n–1) + fibonacci(n–2)
  fi

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

```
if n = 0 → return 0;
 | n = 1 → return 1;
 | n ≥ 2 →  x, y := 0, 1;
      for i:=2 to n do
        fibo, x, y  := x+y,  y, fibo
      od
      return fibo
fi;
```

```
if n = 0 → return 0;
 | n = 1 → return  1;
 | n ≥ 2 →  f0, f1 := 0, 1;
      for i:=2 to n do
        f0, f1  := f1, f0+f1
      od
      return f1
fi;
```


The Merge Sort

function *sort*$(L = l_1, l_2, \ldots, l_n \in \mathbf{N}^n) \in \mathbf{N}^n$;
  if $n > 1 \rightarrow m := \lfloor n/2 \rfloor$; $L = merge(sort(l_1, \ldots, l_m), sort(l_{m+1}, \ldots, l_n))$
  | $n \leq 1 \rightarrow sort := L$;
  fi
    $O(\log n)$

function *merge*$(A = (a_1, \ldots, a_m) \in \mathbf{N}^m; B = (b_1, \ldots, b_n) \in \mathbf{N}^n) \in \mathbf{N}^{m+n}$;
  { Assert $(a_1 \leq a_2 \leq \ldots \leq a_m) \wedge (b_1 \leq b_2 \leq \ldots \leq b_n)$ }
  if $m = 0 \rightarrow merge := B$
  | $n = 0 \rightarrow merge := A$
  elsif | $a_1 \leq b_1 \rightarrow merge := (a_1, merge((a_2, \ldots, a_m); B))$
    | $b_1 \leq a_1 \rightarrow merge := (b_1, merge(A; (b_2, \ldots, b_m)))$
  fi
    $O(n)$

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

L = *empty list;* $i, j, k, := 1, 1, 1;$

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

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

$\mid\ a_i \le b_j \rightarrow L_k := a_i;\ i, k := i+1, k+1$

$\mid\ a_i \ge b_j \rightarrow L_k := b_j;\ j, k := j+1, k+1$

fi

od;

iterative merge $:= L.$

$O(n \log n)$

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

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

fi ***od***

5.5 Program Correctness

Definition *1 A Program S is said to be* ***partially correct****,*
w.r.t the ***initial assertion*** *p and the* ***final assertion*** *q.*

If the ***initial condition*** *p is true and S* ***terminates****, then the final assertion q is true, written* $p\{S\}q$ *and it is called* ***Hoare triple****.*

Ex. 1 $(x=1)\ \{y:=2;\ z:=x+y\}\ (z=3)$

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

Sequence *of statement*

$p\ \{S_1\}\ q$

$q\ \{S_2\}\ r$

$\therefore\ \overline{p\ \{S_1; S_2\}\ r}$

if B ***then*** S ***fi***

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

If B ***then*** S_1 ***else*** S_2

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

Ex. ***if*** $x \geq y$ ***then*** $m:=x$ ***else*** $m:=y$ ***fi***

$T\{$***if*** $x \geq y$ ***then*** $m:=x$ ***else*** $m:=y$ ***fi***$\}(m=x \vee m=y) \wedge (m \geq x) \wedge (m \geq y)$

while B ***do*** S ***od***

$$\frac{(p \wedge B)\,\{S\}\, p}{\therefore\ p\,\{\textbf{while } B \textbf{ do } S\ B \textbf{ od}\}\,(p \wedge \neg B)}$$

loop invariant p

initialize p

update p

terminate $(p \wedge \neg B)$

function $prod(m, n: integer): integer;$ $\quad m, n \in \mathbf{Z}$

if $n < 0$ ***then*** $a := -n$ ***else*** $a := n;$	$a = \lvert n \rvert$
$x, k := 0, 0;$	$x = m \cdot k \quad$ /* $k=0$ */
do $k < a \rightarrow$	$x = m \cdot k$
$x := x+m;$ /* $x = m \cdot (k-1)$ */ $++k$	$x = m \cdot k$
od	$x = m \cdot k \wedge \neg(k<a) = m \cdot k \wedge (k=a) \ \therefore x = ma = m \cdot \lvert n \rvert$
if $n < 0$ ***then*** $prod := -x$ ***else*** $prod := n;$	$prod = m \cdot n$