**Definition 1 – Strong Induction**

Let $i \in \mathbb{N}$ be arbitrary;
Assume $\forall j \in \mathbb{N}.[(j < i)$ IMPLIES $P(j)]$
...
$P(i)$
$\forall i \in \mathbb{N}.P(i)$ by strong induction

Base Case:
...
P(0)
Induction Step:
Let $i \in \mathbb{Z}^+$ be arbitrary
Assume $\forall j \in \mathbb{N}.[(j < i)$ IMPLIES $P(j)]$
...
$P(i)$

**Theorem 1**

For all $n \geq 4$, $n$ can be made using only 2 and 5.

*Proof.* For all $n \in \mathbb{N}$, let $P(n) =$ "$n \geq 4$ IMPLIES $\exists k_1 \in \mathbb{N}.\exists k_2 \in \mathbb{N}.(n = 2k_1 + 5k_2)$"
Let $i \in \mathbb{N}$ be arbitraru
Assume $\forall j \in \mathbb{N}.[(j < i)$ IMPLIES $P(j)]$
If $i = 4$ then $i = 2(2) + 5(0), k_1 = 2, k_2 = 0$
So $i \geq 4$ IMPLIES $\exists k_1 \in \mathbb{N}.\exists k_2 \in \mathbb{N}.(i = 2k_1 + 5k_2)$
If $i = 5$ then $i = 2(0) + 5(1), k_1 = 0, k_2 = 1$
So $P(5)$.
If $i \geq 6$ then $4 \leq i - 2 \leq i$
$\forall j \in \mathbb{N}.(j \leq i)$ IMPLIES $P(j)$ $(i - 2 < i$ IMPLIES $P(i - 2))$
$P(i - 2)$ is true (modus ponens)
thus $\exists k_1' \in \mathbb{N}.\exists k_2' \in \mathbb{N}.(i - 2 = 2k_1' + 5k_2')$
Let $k_1 = k_1' + 1, k_2 = k_2'$, then $i = 2k_1 + 5k_2$
$i \geq 4$ IMPLIES $\exists k_1 \in \mathbb{N}.\exists k_2 \in \mathbb{N}.(i = 2k_1 + 5k_2)$
$\forall i \in \mathbb{N}.i \geq 4$ IMPLIES $P(i)$ proof by strong induction.

QUOD ERAT DEM■

**Theorem 2**

Every integer greater than 1 is a product of prime.

*Proof.*
For all $n \in M = \{n \in \mathbb{N} \mid n > 1\}$,
let $P(n)$="$n$ is a product of primes.
Let $n \in M$ be arbitrary
Assume $\forall j \in M.[(j < n)$ IMPLIES $P(j)]$
If $n$ is prime, then $P(n)$ (already a product of 1 prime).
Otherwise, $n$ is compose so there exist $k \in M$ and $m \in M$ such that $n = k \times m$.
Since $k < n$ and $m < n$, $P(k)$ and $P(m)$, it follows by the induction hypothesis that $K$ is a product of primes and $m$ is a product of primes,
Hence $n = k \times m$ is a product of primes.
$P(n)$
By strong induction $\forall n \in M.P(n)$

QUOD ERAT DEM■

**Definition 2 – Recursively Defined Sets**

2 parts to the Definition

1) A base case that does not depend on anything else
2) a constructor case, that depends on previous cases.

Example:
1. $\{0,1\}^\star$ = set of all fiinte strings of bits
Base case: $\lambda \in \{0,1\}^\star$, $\lambda$ is the empty string which contains no bits.
Constructor Case: If $s \in \{0,1\}^\star$, then $s0 = s \cdot 0$ and $s1 = s \cdot 1$ are in $\{0,1\}^\star$

**Remark 1.** More generally if $\Sigma$ is any finite set of letters, then $\Sigma^\star$ is the set of all finite strings of letters from $\Sigma$

Example: $\Sigma = \{[,]\}$

△

☺

2. Let Bkts = set of finite strings of matched brackets
Base case: $\lambda \in$Bkts.
Constructor Case(s): If $a, t \in$ Bkts, then

1) $[a] \in$ Bkts
2) $at \in$ Bkts

Alternate constructor case: If $a, t \in$ Bkts, then $[a]t \in$ Bkts ☺

3. Syntactically correct formulas of propositional logic $S$.
Base Case: the set of all propositional variables $\subseteq S$.
Constructor Cases: If $f, f' \in S$ then
$(f \text{ AND } f') \in S$
$(f \text{ OR } f') \in S$
$(f \text{ IMPLIES } f') \in S$
$(f \text{ IFF } f') \in S$
NOT$f \in S$
$(f \text{ XOR } f') \in S$

☺

4. $M =$ Syntactically correct monotone formulas of propositional formulas
Base Case: Set of all propositional variables $\subseteq$ M.
Constructor Case: If $f, f' \in M$, then $(f \text{ OR } f') \in M$ and $(f \text{ AND } f') \in M$

**Remark 2.** We mean, but may not say explicitely that we are defining the smallest such set.

△

☺

5. Arthmetic expressions involving natural numbers
Base Case $\mathbb{N} \subseteq A$
Constructor Case: If $e, f \in A$, then $e + f \in A$ and $e \times f \in A$.

**Remark 3.** $2 \times 3 + 1$ is ambiguous. When a recursive definition allows an element to be constructed in more than one way, the definition is ambiguous

△

☺

**Definition 3 – Structural Induction**

A form of induction used to prove properties about recursively defined set.

Let $S$ be a recursively defined set, consider the monotone formulas of propositional logic $M$.
To prove $\forall x \in S.P(x)$ where $p : S \to \{T, F\}$ is a predicate.
prove $p(x)$ for all base cases of the definition, $p(x)$ for the constructor cases of the definition, assuming $P$ is true for the components of $x$.
For all $f \in M$, let
$N_v(f)$= # of occurences of propositional variables in $f$
$N_c(f)$= # of occurences of connectives in $f$
$p(f)$ = "$N_v(f) = 1 + N_c(f)$"
Base Case: if $f$ is a propositional variable, then $N_v(f) = 1 = 1 + 0 = 1 + N_c(f)$, $p(f)$ is True.
Constructor Case:
let $f', f'' \in M$ be arbitrary
Assume $p(f')$ and $p(f'')$) are True,
Consider $f = (f' \star f'')$ where $\star \in \{$ AND , OR $\}$, then $N_v(f) = N_v(f') + N_v(f'') = (1N_c(f')) + (1 + N_c(f'')) = 1 + (N_c(f') + N_c(f'') + 1) = 1 + N_c(f)$

☺

**Remark 4.** $\mathbb{N}$ can be defined recursively
Base case: $0 \in \mathbb{N}$
Constructor case: if $n \in \mathbb{N}$, then $n + 1 \in \mathbb{N}$
So weak induction can be viewed as stuctural induction.

△

To prove $\forall m \in \mathbb{N}.\forall n \in \mathbb{N}.p(m, n)$, which is equivalent to $\forall (m, n) \in \mathbb{N} \times \mathbb{N}.p(m, n)$ where $\mathbb{N} \times \mathbb{N}$ can be defined recursively:
Base case $(0, 0) \in \mathbb{N} \times \mathbb{N}$
Constructor Case: if $(m, n) \in \mathbb{N} \times \mathbb{N}$, then $(m + 1, n) \in \mathbb{N} \times \mathbb{N}$ and $(m, n + 1) \in \mathbb{N} \times \mathbb{N}$.
To prove $\forall (m, n) \in \mathbb{N} \times \mathbb{N}.p(m, n)$ using Structural induction:
Base Case: $p(0, 0)$
Constructor Case: let $(m, n) \in \mathbb{N} \times \mathbb{N}$ be arbitrary,
Assume $p(m, n)$, then prove $p(m, n + 1)$ and $p(m + 1, n)$, after that we can conclude $\forall (m, n) \in \mathbb{N} \times \mathbb{N}.p(m, n)$

**Remark 5.** Here we are only assuming "one block" and need to prove 2.

△

☺

**Definition 4 – Strong Induction alternative**

Let $(m, n) \in \mathbb{N} \times \mathbb{N}$ be arbitrary.
Assume $\forall (i, j) \in \mathbb{N} \times \mathbb{N}.[(i < m \text{ AND } j \leq n) \text{ OR } (i \leq m \text{ AND } j < m)]$ IMPLIES $p(i, j)$
...
$p(m, n)$

**Remark 6.** Here we are assuming all blocks except the bottom right one, and want to prove this one.

△

**Definition 5 – Double Indunction (one induction inside another induction)**

Define $Q(m) = \forall n \in \mathbb{N}.p(m, n)$,
To prove $\forall m \in \mathbb{N}.\forall n \in \mathbb{N}.p(m, n)$ is same as proving $\forall m \in \mathbb{N}.Q(m)$.
Let $m \in \mathbb{N}$ be arbitrary;
Assume $\forall i \in \mathbb{N}.(i < m)$ IMPLIES $Q(i)$.
Let $n \in \mathbb{N}$ be arbitrary;
Assume $\forall j \in \mathbb{N}.(j < n)$ IMPLIES $p(m, j)$
...
$p(m, n)$
$\forall n \in \mathbb{N}.p(m, n)$
$\forall m \in \mathbb{N}.Q(m)$
$\forall m \in \mathbb{N}.\forall n \in \mathbb{N}.p(m, n)$

**Remark 7.** Here we are assuming all the blocks above the line of the current block, and all blocks at the left of the current block, and want to prove the current block. (better than the previous 2..?)

△

Defining functions on recursively defined sets.
Exmaple 1.
$N_v : M \to \mathbb{N}$,
$N_v = 1$ for any propositional variables
$N_v = N_v(f') + N_v(f'')$ for $f \in f' \star f''$ where $\star \in \{ \text{ AND }, \text{ OR } \}$
Example 2.
value:$A \to \mathbb{N}$
for $f \in A$, let value$(f)$ be the natural number represented by $f$
Base case: value$(a)$= $a$ for all $a \in \mathbb{N}$
Constructor cases:
value$(f + g)$=value$(f)$+value$(g)$
value$(f \times g)$=value$(f)\times$value$(g)$

**Remark 8.** Consider $2 \times 3 + 1$, the value is undefined, thus this example is ambiguous. Be careful when defining functions on recursively defined sets that are ambiguous.

△

Example 3.
Let $B$ be the set of all binary trees,
Base case: empty tree $\in B$.
Constructor case: if $t_1, t_2 \in B$, and $r$ is a node, then $r(t_1, t_2) \in B$ (node $r$ with $t_1$ left child and $t_2$ right child), and $t_1$ =left($t$) and $t_2$ =right($t$).
Example 4.
N:$B \to \mathbb{N}$ (number of nodes)
Base case: $N$(empty tree)=0
Constructor case: $N$(t) = 1 + $N$(left($t$)) + $N$(right($t$))
Then $N(t)$= # of nodes in $t$.
**Example 5.**
$L : B \to \mathbb{N}$
Base Case $L$(empty tree)=0
Constructor Case $L(t) = L(\text{left}(t)) + L(\text{right}(t))$
$L(t)$ = # of leaves in $t$.
Note $L(t) = 0$ for all $t \in B$.
**this is a false example**
thus we need another base case $L$(one node tree)=1 ☺

**Theorem 3**

A binary tree with $n$ node has at most $\lceil \frac{n}{2} \rceil$ leaves
Let $q : B \to \{T, F\}$ ($B$ is the set of binary trees)
be $q(t)$="$L(t) \leq \lceil \frac{N(t)}{2} \rceil$"
$\forall t \in B.q(t)$

*Proof.* For $t \in B$ and $n \in \mathbb{N}$, let $S(t, n)$ ="$t$ has $n$ nodes" and $Al(t, n)$ ="$t$ has at most $n$ leaves".
Let $p(n) = \forall t \in B.[S(t, n)$ IMPLIES $Al(t, \lceil \frac{n}{2} \rceil)]$.

QUOD ERAT DEM■