String definitions and induction

**Strings**

**Recursive definition of Strings:** Let $A$ be a non-empty set of characters (or letters, symbols). $A$ is called an alphabet. The set of strings over alphabet $A$, denoted $A^*$ is defined as follows.

- **Base Case:** The empty string $\lambda$ is in $A^*$.
- **Constructor Case:** If $a \in A$ and $s \in A^*$ then $\langle a, s \rangle \in A^*$.

**Length of Strings:** Length $|s|$ of a string $s$ is defined recursively as

- **Base Case:** $|\lambda|$ is defined to be 0
- **Constructor Case:** $|(a, s)|$ is $1 + |s|$.

**Concatenation:** The concatenation of string $s$ with $t$, denoted $s \cdot t$ is recursively defined as

- **Base Case:** $\lambda \cdot t$ is $t$
- **Constructor Case:** $\langle a, s \rangle \cdot t$ is $\langle a, s \cdot t \rangle$.

**Proposition 1.** $s \cdot \lambda = s$ for all $s \in A^*$.

**Proposition 2.** For all $s, t \in A^*$, $|s \cdot t| = |s| + |t|$. 
Structural Induction: Let $P$ be a predicate on a recursively defined data type $R$. If

- $P(b)$ is true for each base case element $b \in R$, and
- for all $k$-argument constructors $c$
  
  
  $[P(r_1) \text{ AND } P(r_2) \text{ AND } \cdots \text{ AND } P(r_k)] \text{ IMPLIES } P(c(r_1, r_2, \ldots, r_k))$

  for all $r_1, r_2, \ldots, r_k \in R$
then $P(r)$ is true for all $r \in R$.

Well matched Brackets

Definition: The set of well-match strings, RecMatch, can be defined as

- **Base Case:** $\lambda \in \text{RecMatch}$
- **Constructor Case:** If $s, t \in \text{RecMatch}$ then $\langle[,\lambda] \cdot s \cdot [,\lambda]\rangle \cdot t \in \text{RecMatch}$.

Number of characters: $\#_c(s)$ is the number of occurrences of $c$ in $s$, and can be defined recursively as

- **Base Case:** $\#_c(\lambda) = 0$
- **Constructor Case:** $\#_c(\langle a, s \rangle) = \#_c(s)$ if $a \neq c$, and $\#_c(\langle a, s \rangle) = 1 + \#_c(s)$ if $a = c$.

Proposition 3. Every string in RecMatch has an equal number of $[$ and $]$ symbols.