Structural Induction

Definition The set of RNA strands S is defined (recursively) by:

Basis Step:	$\mathtt{A} \in S, \mathtt{C} \in S, \mathtt{U} \in S, \mathtt{G} \in S$
Recursive Step:	If $s \in S$ and $b \in B$, then $sb \in S$

where sb is string concatenation.

The function *rnalen* that computes the length of RNA strands in S is defined recursively by $rnalen : S \to \mathbb{Z}^+$

Basis step: If $b \in B$ then $rnalen(b) = 1$

Recursive step: If $s \in S$ and $b \in B$, then $rnalen(sb) = 1 + rnalen(s)$

The function *basecount* that computes the number of a given base b appearing in a RNA strand s is defined recursively by $basecount : S \times B \to \mathbb{N}$

Basis step: If $b_1 \in B$, $b_2 \in B$, $basecount(b_1, b_2) = \begin{cases} 1 & \text{when } b_1 = b_2 \\ 0 & \text{when } b_1 \neq b_2 \end{cases}$

Recursive Step: If $s \in S$, $b_1 \in B$, $b_2 \in B$, $basecount(sb_1, b_2) = \begin{cases} 1 + basecount(s, b_2) & \text{when } b_1 = b_2 \\ basecount(s, b_2) & \text{when } b_1 \neq b_2 \end{cases}$

Prove or disprove $\exists s \in S\ (rnalen(s) = basecount(s, \mathtt{A}))$:

Prove or disprove $\forall s \in S\ (rnalen(s) \geq basecount(s, \mathtt{A}))$:

Proof:

Basis case: Assume $s = \mathtt{A} \vee s = \mathtt{C} \vee s = \mathtt{U} \vee s = \mathtt{G}$. Need to show $rnalen(s) \geq basecount(s, \mathtt{A})$.

Case 1: Want to show $(s = \mathtt{A}) \to (\ rnalen(s) \geq basecount(s, \mathtt{A})\ )$.

Case 2: Want to show $(s = \mathtt{C} \vee S = \mathtt{U} \vee S = \mathtt{G}) \to (\ rnalen(s) \geq basecount(s, \mathtt{A})\ )$.

Continued next page

Proof by universal generalization: To prove that $\forall x\, P(x)$ is true, we can take an arbitrary element e from the domain and show that $P(e)$ is true, without making any assumptions about e other than that it comes from the domain.

New! Proof by Structural Induction (Rosen 5.3 p354) To prove a universal quantification over a recursively defined set:

> **Basis Step**: Show the statement holds for elements specified in the basis step of the definition.
>
> **Recursive Step**: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

Recursive case: Want to show

$$\forall e \in S \ (\ \mathit{rnalen(e)} \geq \mathit{basecount}(e, \mathtt{A}) \rightarrow \forall b \in B \ (\ \mathit{rnalen(eb)} \geq \mathit{basecount}(eb, \mathtt{A}) \ ) \ )$$

Consider arbitrary e. Assume, as the **induction hypothesis** that

$$\mathit{rnalen(e)} \geq \mathit{basecount}(e, \mathtt{A})$$

Need to show

$$\forall b \in B \ (\ \mathit{rnalen(eb)} \geq \mathit{basecount}(eb, \mathtt{A}) \ )$$

Consider arbitrary $b \in B$.

Case 1: Want to show $(b = \mathtt{A}) \rightarrow (\ \mathit{rnalen(eb)} \geq \mathit{basecount}(eb, \mathtt{A}) \ )$.

Case 2: Want to show $(b = \mathtt{C} \vee b = \mathtt{U} \vee b = \mathtt{G}) \rightarrow (\ \mathit{rnalen(eb)} \geq \mathit{basecount}(eb, \mathtt{A}) \ )$.

Proof by Structural Induction (Rosen 5.3 p354) To prove a universal quantification over a recursively defined set:

> **Basis Step**: Show the statement holds for elements specified in the basis step of the definition.
>
> **Recursive Step**: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

Definition The set of natural numbers (aka nonnegative integers), $\mathbb{N}$, is defined (recursively) by:

Basis Step:	$0 \in \mathbb{N}$
Recursive Step:	If $n \in \mathbb{N}$ then $n + 1 \in \mathbb{N}$ (where $n + 1$ is integer addition)

The function *sumPow* with domain $\mathbb{N}$, codomain $\mathbb{N}$, and which computes, for input i, the sum of the first i powers of 2 is defined recursively by $sumPow : \mathbb{N} \to \mathbb{N}$ with

Basis step: $sumPow(0) = 1$.

Recursive step: If $x \in \mathbb{N}$ then $sumPow(x + 1) = sumPow(x) + 2^{x+1}$.

Fill in the blanks in the following proof of $\forall n \in \mathbb{N}\, (sumPow(n) = 2^{n+1} - 1)$:

Since $\mathbb{N}$ is recursively defined, we proceed by ____________________.

Basis case We need to show that ____________________. Evaluating each side: $LHS = sumPow(0) = 1$ by the basis case in the recursive definition of *sumPow*; $RHS = 2^{0+1} - 1 = 2^1 - 1 = 2 - 1 = 1$. Since $1 = 1$, the equality holds.

Recursive step Consider arbitrary natural number n and assume, as the ____________________ that $sumPow(n) = 2^{n+1} - 1$. We need to show that ________________________. Evaluating each side:

$$LHS = sumPow(n+1) \overset{\text{rec def}}{=} sumPow(n) + 2^{n+1} \overset{\text{IH}}{=} (2^{n+1} - 1) + 2^{n+1}.$$

$$RHS = 2^{(n+1)+1} - 1 \overset{\text{exponent rules}}{=} 2 \cdot 2^{n+1} - 1 = \left(2^{n+1} + 2^{n+1}\right) - 1 \overset{\text{regrouping}}{=} (2^{n+1} - 1) + 2^{n+1}$$

Thus, $LHS = RHS$. The structural induction is complete and we have proved the universal generalization.

Extra example Connect the function *sumPow* to binary expansions of positive integers.

Definition The set of linked lists of natural numbers L is defined:

Basis Step:	$[] \in L$
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$, then $(n, l) \in L$

Examples:

Definition The length of a linked list of natural numbers L, $length : L \to \mathbb{N}$ is defined by:

Basis Step:		$length([])$	$= 0$
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$, then	$length((n, l))$	

Examples:

Extra example: The function $prepend : L \times \mathbb{N} \to L$ that adds an element at the front of a linked list is defined:

Definition The function $append : L \times \mathbb{N} \to L$ that adds an element at the end of a linked list is defined:

Basis Step:	If $m \in \mathbb{N}$ then
Recursive Step:	If $l \in L$ and $n \in \mathbb{N}$ and $m \in \mathbb{N}$, then

Examples:

Claim: $\forall l \in L\ (\ length(\ append(l, 100)\ ) > length(l)\ )$

Proof: By structural induction on L, we have two cases:

Basis Step

1.	**To Show** $length(\ append([], 100)\ ) > length([])$	Because [] is the only element defined i step of L, we only need to prove that th holds for [].
2.	**To Show** $length(\ (100, [])\ ) > length([])$	By basis step in definition of *append*.
3.	**To Show** $(1 + length([])) > length([])$	By recursive step in definition of *lengt*
4.	**To Show** $1 + 0 > 0$	By basis step in definition of *length*.
5.	**To Show** T	By properties of integers
	QED	Because we got to T only by rewriting to equivalent statements, using well-de techniques, and applying definitions.

Recursive Step

Consider an arbitrary: $l = (n, l')$, $l' \in L$, $n \in \mathbb{N}$, and we assume as the **induction hypothesis** that:

$$length(\ append(l', 100)\ ) > length(l')$$

Our goal is to show that $length(\ append((n, l'), 100)\ ) > length((n, l'))$ is also true. We evaluate each side of the candidate inequality:

$LHS = length(\ append((n, l'), 100)\ ) = length(\ (n, append(l', 100))\ )$ by the recursive definitio

$= 1 + length(\ append(l', 100)\ )$ by the recursive definition of *length*

$> 1 + length(l')$ by the induction hypothesis

$= length((n, l'))$ by the recursive definition of *length*

$= RHS$