HW5 Collaborative

CS40 Spring '22

Due: Friday, May 13, 2022 at 11:59PM on Gradescope

In this assignment,

You will work with recursively defined sets and functions and prove properties about them, practicing induction and other proof strategies.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "HW5-Collaborative".

In your proofs and disproofs of statements below, justify each step by reference to the proof strategies we have discussed so far, and/or to relevant definitions and calculations. We include only induction-related strategies here; you can and should refer to past material to identify others.

Proof by Structural Induction: To prove that $\forall x \in X\, P(x)$ where X is a recursively defined set, prove two cases:

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.

Proof by Mathematical Induction: To prove a universal quantification over the set of all integers greater than or equal to some base integer b:

Basis Step:	Show the statement holds for b.
Recursive Step:	Consider an arbitrary integer n greater than or equal to b, assume (as the **induction hypothesis**) that the property holds for n, and use this and other facts to prove that the property holds for $n + 1$.

Proof by Strong Induction To prove that a universal quantification over the set of all integers greater than or equal to some base integer b holds, pick a fixed nonnegative integer j and then:

Basis Step:	Show the statement holds for b, $b + 1$, ..., $b + j$.
Recursive Step:	Consider an arbitrary integer n greater than or equal to $b + j$, assume (as the **strong induction hypothesis**) that the property holds for **each of** b, $b + 1$, ..., n, and use this and other facts to prove that the property holds for $n + 1$.

RNA related definitions

Consider the following definitions related to RNA strands:

Definition Set of bases $B = \{\texttt{A}, \texttt{C}, \texttt{U}, \texttt{G}\}$. The set of RNA strands S is defined (recursively) by:

$$\begin{array}{ll} \text{Basis Step:} & \texttt{A} \in S, \texttt{C} \in S, \texttt{U} \in S, \texttt{G} \in S \\ \text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B, \text{ then } sb \in S \end{array}$$

where sb is string concatenation.

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

$$\begin{array}{ll} \text{Basis Step:} & \text{If } b \in B, \text{ then } rnalen(b) = 1 \\ \text{Recursive Step:} & \text{If } s \in S \text{ and } b \in B, \text{ then } rnalen(sb) = 1 + rnalen(s) \end{array}$$

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

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

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

Definition The function $mutate : S \times B \to S$ is defined recursively as:

$$\text{Basis step: If } b_1 \in B \text{ and } b_2 \in B,\ mutate(b_1, b_2) = \begin{cases} b_2 & \text{when } b_1 = \texttt{A} \\ b_1 & \text{when } b_1 \neq \texttt{A} \end{cases}$$

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

Linked list related definitions

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

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

Definition The function *removeTail* $: L \to L$ that removes the last node of a linked list (if it exists) is defined by:

$$\begin{array}{ll} & removeTail : L \to L \\ \text{Basis Step:} & removeTail([\,]) = [\,] \\ \text{Recursive Step:} & \text{If } l \in L \text{ and } n \in \mathbb{N}, \text{ then} \\ & removeTail(\ (n, l)\ ) = \begin{cases} [\,], \text{when } l = [\,] \\ (n, removeTail(l)\ ), \text{when } l \neq [\,] \end{cases} \end{array}$$

Definition The function *remove* $: L \times \mathbb{N} \to L$ that removes a single node containing a given value (if present) from a linked list is defined by:

$$\begin{array}{llll} & & remove : L \times \mathbb{N} & \to L \\ \text{Basis Step:} & \text{If } m \in \mathbb{N} \text{ then} & remove([\,], m) & = [\,] \\ \text{Recursive Step:} & \text{If } l \in L, n \in \mathbb{N}, m \in \mathbb{N}, \text{ then} & remove((n, l), m) & = \begin{cases} l & \text{when } n = m \\ (n, remove(l, m)) & \text{when } n \neq m \end{cases} \end{array}$$

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

$$prepend(l, n) = (n, l)$$

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

$$\begin{array}{llll} & & append : L \times \mathbb{N} & \to L \\ \text{Basis Step:} & \text{If } m \in \mathbb{N} \text{ then} & append([\,], m) & = (m, [\,]) \\ \text{Recursive Step:} & \text{If } l \in L \text{ and } n \in \mathbb{N} \text{ and } m \in \mathbb{N}, \text{ then} & append((n, l), m) & = (n, append(l, m)) \end{array}$$

Assigned Questions

Definitions related to RNA and linked list are listed on the previous two pages.

1. (*Graded for correctness*) Calculate the following function applications. Include all intermediate steps, with justifications.

   Sample response that can be used as reference for the detail expected in your answer for this part:

   Calculating *append*($(1,(2,[\,]))$,3), we have

   $$\begin{aligned} append(\ (1,(2,[\,]))\ ,3) &= (1, append(\ (2,[\,])\ ,3)) && \text{By recursive step of } append\text{: } n=1,\ l=(2,[\,]),\ m=3 \\ &= (1,\ (2,\ append(\ [\,]\ ,3))) && \text{By recursive step of } append\text{: } n=2,\ l=[\,],\ m=3 \\ &= (1,\ (2,\ (3,[\,]))) && \text{By basis step of } append\text{: } m=3 \end{aligned}$$

   (a) Calculate $removeTail(\ append(\ (2,(3,[\,])\ ),1\ )\ )$
   Answer:
   $removeTail(\ append(\ (2,(3,[\,])\ ),1\ )\ )$ Base statement
   $removeTail(\ (2,\ append((3,[\,]),1\ )\ )\ )$ By recursive step of append, n=2, l=(3,[]), m=1
   $removeTail(\ (2,3\ append(([\,]),1\ )\ )\ )$ By recursive step of append, n=3, l=([]), m=1
   removeTail((2,3,1,[])) By basis step of append, m=1
   $(2, removeTail(\ (3,1,[\,]\ )\ )\ )$ By recursive step of removeTail, m=(2,3,1,[])
   $(2,3, removeTail(\ (1,[\,]\ )\ )\ )$ By recursive step of removeTail, m=(3,1,[])
   (2,(3,[])) By basis step of removeTail, m=[]

   (b) Calculate $prepend(\ remove(\ (1,(2,(2,(3,[\,])))),2\ ),3\ )$
   prepend(remove((1, (2, (2, (3,[])))), 2) , 3) Base statement
   prepend((1, remove((2, (2, (3,[])))), 2) , 3) By recursive step of remove, m=(2, (2, (3,[])))
   prepend((1, (2, (3,[]))) , 3) By base step of remove, m=(2, (3,[]))
   $(3,(1,(2,(3,[\,]))))$ By base step of prepend, m=(3,(1, (2, (3,[]))))

2. Consider the following statement and attempted proof:

$$\forall l \in L\, \exists n \in \mathbb{N}\ (\ append(removeTail(l), n) = l\ )$$

Attempted proof: By structural induction on L, we have two cases:

Basis Step: Consider $l = []$ and choose the witness $n = 0$ (in the domain $\mathbb{N}$ since it is a nonnegative integer). We need to show that $append(removeTail(\ (0, [])\ ), 0) = (0, [])$. By the definition of $removeTail$, using the recursive step with $l = []$ and $n = 0$, we have $removeTail(\ (0, [])\ ) = []$. By the definition of $append$, using the basis step with $l = []$ and n, we have $append([], 0) = (0, [])$ as required.

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

$$\exists n \in \mathbb{N}\ (\ append(removeTail(l'), n) = l'\ )$$

Our goal is to show that $\exists n \in \mathbb{N}\ (\ append(removeTail((x, l')), n) = (x, l')\ )$. Choose the witness $n = x$, a nonnegative integer so in the domain. We need to show that

$$append(removeTail((x, l')), x) = (x, l')$$

Applying the definitions:

$$\begin{aligned} LHS &= append(removeTail((x, l')), x) \\ &= append((x, removeTail(l')), x) \qquad \text{by recursive step of } removeTail\text{, with } l = l' \text{ and } n = x \\ &= (x, l') \qquad \text{by the recursive definition of } append\text{, with } l = l' \text{ and } n = x \\ &= RHS \end{aligned}$$

as required.

Thus, the recursive step is complete and we have finished the proof by structural induction. ■

(a) (*Graded for correctness*) Demonstrate that this attempted proof is invalid by providing and justifying a **counterexample** (disproving the statement).
Solutions: Proof by counterexample.
Take $x = 1$ and $l' = (2, [])$ so that $append(removeTail((1, (2, []))), 1) = (1, (2, []))$
Applying definitions, $LHS = append((1, []), 1) = (1, (1, [])) \neq (1, (2, [])) = RHS$
$LHS \neq RHS$ ■
This is clearly false, proving the statement false through counterexample.

(b) (*Graded for fair effort completeness*) Explain why the attempted proof is invalid by identifying in which step(s) a definition or proof strategy is used incorrectly, and describing how the definition or proof strategy was misused.
This proof assumes that the tail element of L and the element n will have the same value. In fact, this statement holds true under the condition that the element being removed is the same as the one being appended. To create a statement that doesn't depend on this condition and holds a similar meaning, we can simply swap the order of append and removeTail. This ensures removeTail removes whatever appends adds, preserving any l for any value n.

3. (*Each part graded for correctness in evaluating statement and for fair effort completeness in the justification)* Statements like these are used to build the specifications for programs, libraries, and data structures (API) which spell out the expected behavior of certain functions and methods. In this HW question, you're analyzing whether and how order matters for the *remove* and *prepend* functions.

   (a) Prove or disprove the following statement:

$$\forall l \in L\, \forall m \in \mathbb{N}\ (\ prepend(\ remove(l, m)\ ,\ m) = l\ ).$$

   Solutions: Disproof by counterexample.
   Take $l = (1, (2, (3, [\,])))$ and $m = 4$, the statement evaluates to:

$$\begin{aligned}
LHS &= prepend(\ remove(\ (1, (2, (3, [\,])))\ , 4)\ ,\ 4) \\
&= prepend(\ (1, (2, (3, [\,])))\ , 4\ ) \\
&= (4, (1, (2, (3, [\,])\ )\ )\ ) \\
&\neq (1, (2, (3, [\,]))) = l \\
LHS &\neq RHS
\end{aligned}$$

   (b) Prove or disprove the following statement:

$$\exists l \in L\, \exists m \in \mathbb{N}\ (\ prepend(\ remove(l, m)\ , m) = l\ ).$$

   Answer: Unlike the previous question that used two $\forall$ quantifiers, this question uses two $\exists$ quantifiers. Already, this statement looks like it has better chances of being true. Let's try to prove the statement with existiential generalization. For l=[1,2,3] and m=1, the statement equals: *Solutions:* Proof by existential generalization, proof by witness.
   Take $l = (1, (2, (3, [\,])))$ and $m = 1$, the statement evaluates to:

$$\begin{aligned}
LHS &= prepend(\ remove(\ (1, (2, (3, [\,])))\ , 1)\ ,\ 1) \\
&= prepend(\ (2, (3, [\,]))\ , 1\ ) \\
&= (1, (2, (3, [\,])\ )\ ) \\
&= (1, (2, (3, [\,])\ )\ ) = l \\
LHS &= RHS
\end{aligned}$$

   This expression is true, proving there exists some values l and m that cause the expression $(\ prepend(\ remove(l, m)\ , m) = l\ )$ to be true. This proves the original statement through existiential generalization. Interestingly, the proposition within the quantifiers will evaluate to true so long as m equals the first (head) element of l.

4. Write the first 6 terms of the sequence that is described by each of the recurrence relations below:

   (a) $f_1 = 1, f_2 = 3$, and $f_n = 5f_{n-1} - 2f_{n-2}$ for $n \geq 3$.
   Answer: 1, 3, 13, 59, 269, 1227, 5597, 25531

   (b) $g_1 = 4$ and $g_2 = 5$. The rest of the terms are given by the formula $g_n = ng_{n-1} + g_{n-2}$.
   Answer: 4, 5, 19, 81, 424, 2625

5. (*Graded for correctness*) Prove that

$$\exists n_0 \in \mathbb{N}\, \forall n \in \mathbb{Z}^{\geq n_0}\ (\ n^3 \leq (n+2)!\ )$$

Answer: By the constraints, n must be greater or equal to zero. Let's establish the base case for n=0:
Base Case: $(0^3 \leq (0+2)!)$
$(0 \leq 1 * 2)$
$(0 \leq 2)$
The base case is true.
Assume that by inductive hypothesis:
$((n+1)^3 \leq ((n+1)+2)!)$
$((n+1)^3 \leq (n+3)!)$ *Algebra*
$((n+1)^3 \leq ((n+3)(n+2)(n+1)(n)!)$ *Algebra*
$(1 \leq \frac{(n+3)(n+2)(n)!}{(n+1)(n+1)})$ Divide both sides
$\frac{(n+3)(n+2)}{(n+1)(n+1)}$ is greater than 1 for any values of n. n! is greater or equal to 1 for any values of n. Therefore, this statement is true. By proving the base and inductive steps, we prove this proposition true through induction.

6. (*Graded for correctness in evaluating statement and for fair effort completeness in the justification*) Consider the functions $f_a : \mathbb{N} \to \mathbb{N}$ and $f_b : \mathbb{N} \to \mathbb{N}$ defined recursively by

$$f_a(0) = 0 \qquad \text{and for each } n \in \mathbb{N}, \qquad f_a(n+1) = f_a(n) + 2n + 1$$

$$f_b(0) = 0 \qquad \text{and for each } n \in \mathbb{N}, \qquad f_b(n+1) = 2f_b(n)$$

Which of these two functions (if any) equals 2^n and which of these functions (if any) equals n^2? Use induction to prove the equality or use counterexamples to disprove it.
Answer: First we will look at f_a. We will disprove the claims that $f_a = n^2$ and $f_a = 2^n$ with counterexamples.
We start with a counterexample to disprove $f_a = n^2$.
For n $= 2$, $f_a(0) = 0, f_a(1) = 2 * 0 + 1 = 1, f_a(2) = 2 * 1 + 1 = 3$
$3 = 2^2$
$3 = 4$
This statement is not true so we have proven $f_a = n^2$ is false through counterexample.
We move onto a counterexample to disprove $f_a = 2^n$.
For n $= 2$, $f_a(0) = 0, f_a(1) = 2 * 0 + 1 = 1, f_a(2) = 2 * 1 + 1 = 3$
$3 = 2^2$
$3 = 4$
This statement is not true so we have proven $f_a = n^2$ is false through counterexample. Conveniently enough, $2^n = n^2$ for n=2 so we can actually just use the same calculations twice.
We will look at f_b. We will disprove the claims that $f_b = n^2$ and $f_b = 2^n$ with counterexamples.
We start with a counterexample to disprove $f_b = n^2$.
For n $= 3$, $f_b(0) = 0, f_b(1) = 2 * 0, f_b(2) = 2 * 0, f_b(3) = 2 * 0$
0=3^2
0=9
This statement is not true so we have proven $f_b = n^2$ is false through counterexample. In fact, f_b is always equal to zero for any value n because the base case is zero and any value multiplied by zero will always be zero. With this in mind, we can pretty much use the same calculations to disprove the other claim.
We now use a counterexample to disprove $f_b = 2^n$.
For n $= 3$, $f_b(0) = 0, f_b(1) = 2 * 0, f_b(2) = 2 * 0, f_b(3) = 2 * 0$
0=2^3
0=8
It once again results in a false conclusion, proving the original statement is false through counterexample.

7. Prove that any amount of postage worth 24 cents or more can be made from 7-cent or 5-cent stamps.
Hint: Use mathematical induction
Answer: We will prove this with induction.
Base Step: Find 5 consecutive values
24=2(7)+2(5)
25=5(5)
26=3(7)+1(5)
27=1(7)+4(5)
28=4(7)
We have shown that we can form 5 consecutive values greater or equal than 24 using 5-cent and 7-cent stamps.
Inductive Step: If we can make i cents worth of stamps and make i+1, i+2, i+3, i+4 cents worth of stamps with just 5-cent and 7-cent stamps, we can make any amount of cents worth of stamps as long as it is greater than i by just adding 5-cent coins to whichever combination matches from the base step.
Lemma A: Any value of n$\in Z$ greater than 28 can be reached by adding multiples of 5 to 24,25,26,27,or 28.
We will prove this Lemma by proving it for all its cases:
$\forall n \in Z, \exists i \in Z, (n > 28) \rightarrow (n = 5*i+24 \vee n = 5*i+25 \vee n = 5*i+26 \vee n = 5*i+27 \vee n = 5*i+28)$
We can assume that $\forall n \in Z$, n mod 5 equals 0,1,2,3,4.
Therefore, we can more precisely assume that $\forall n \in Z, n > 28$, n mod 5 equals 0,1,2,3,4.
Case 1: n mod 5 = 0
n = 5i for some i$\in$ Z
25 = 5j for some j$\in$ Z
$\exists k \in Z$, i > j $\rightarrow$ i=j+k
n=5(j+k)=5j+5k=25+5k
For all integers of n where n mod 5 = 0, n can be represented as 25+5k for some integer k that is greater than 28. In other words, they can be formed by adding multiples of 5 to 25.
Case 2: n mod 5 = 1
n = 5i+1 for some i$\in$ Z
26 = 5j+1 for some j$\in$ Z
$\exists k \in Z$, i > j $\rightarrow$ i=j+k
n=5(j+k)+1=5j+5k+1=26+5k
For all integers of n where n mod 5 = 1, n can be represented as 26+5k for some integer k that is greater than 28. In other words, they can be formed by adding multiples of 5 to 26.
Case 3: n mod 5 = 2
n = 5i+2 for some i$\in$ Z
27 = 5j+2 for some j$\in$ Z
$\exists k \in Z$, i > j $\rightarrow$ i=j+k
n=5(j+k)+2=5j+5k+2=27+5k
For all integers of n where n mod 5 = 2, n can be represented as 27+5k for some integer k that is greater than 28. In other words, they can be formed by adding multiples of 5 to 27.
Case 4: n mod 5 = 3
n = 5i+3 for some i$\in$ Z
28 = 5j+3 for some j$\in$ Z
$\exists k \in Z$, i > j $\rightarrow$ i=j+k
n=5(j+k)+3=5j+5k+3=28+5k
For all integers of n where n mod 5 = 3, n can be represented as 28+5k for some integer k that is

greater than 28. In other words, they can be formed by adding multiples of 5 to 28.
Case 5: n mod 5 = 4
n = 5i+4 for some i$\in$ Z
24 = 5j+4 for some j$\in$ Z
$\exists k \in Z$, i > j $\rightarrow$ i=j+k
n=5(j+k)+4=5j+5k+4=24+5k
For all integers of n where n mod 5 = 4, n can be represented as 24+5k for some integer k that is greater than 28. In other words, they can be formed by adding multiples of 5 to 24.

By proving the Base Step and this Lemma A, we have proven that there are 5 consecutive values that can be constructed from 7-cent or 5-cent stamps and that any cent value greater than 28 can be constructed by adding multiples of 5 to these values. This proves the original statement.

Attributions