CSE 20
DISCRETE MATH

Fall 2020

http://cseweb.ucsd.edu/classes/fa20/cse20-a/

Learning goals

Today's goals

- Practice with properties of recursively defined sets and functions
- Define linked lists: a recursively defined data structure
- Prove and disprove properties of recursively defined sets and functions with structural induction

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.

Recursive definitions: recap

We have recursive definitions of

the set of RNA strands

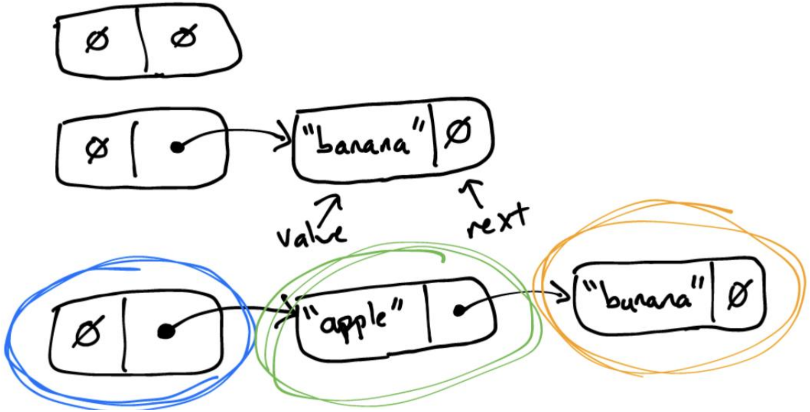
the set of nonnegative integers

the set of integers

What else?

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

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



[Courtesy CSE 12 Winter 2019, Joe Politz]

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

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

Which of the following is a linked list of natural numbers?

A: ( )
B: (3, 7, 9)
C: (9, (3, 7))
D: ([])
E: None of the above

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

$$\begin{array}{lll} \text{Basis Step:} & & length([]) \quad = 0 \\ \text{Recursive Step:} & \text{If } l \in L \text{ and } n \in \mathbb{N}, \text{ then} & length((n, l)) \end{array}$$

How should we fill in the recursive step of the rule?

A: n+1
B: n+l
C: n+length(l)
D: 1+length(l)
E: None of the above

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

$$append(\ [], 1\ ) = (1, [])$$

$$append(\ (1, []), 2\ ) = (1, (2, [])) \quad append(\ (1, (2, [])), 3\ ) = (1, (2, (3, [])))$$

How should we fill in the basis step of the rule?

A: append(m) = []
B: append([]) = m
C: append([],m) = ([],m)
D: append([],m) = (m,[])
E: None of the above

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

$$append(\ [], 1\ ) = (1, [])$$

$$append(\ (1, []), 2\ ) = (1, (2, [])) \quad append(\ (1, (2, [])), 3\ ) = (1, (2, (3, [])))$$

How should we fill in the recursive step of the rule?

A: append(l, m) = (m, l)
B: append((n, l), m) = (n, append(l, m))
C: append((n, l), m) = (m, append(l, n))
D: append((n, l), m) = (l, append(n, m))
E: None of the above

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

Analogy: unit tests in programming

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

Analogy: unit tests in programming

To prove a universal quantification where the element comes from a recursively defined set, consider an arbitrary element and prove two cases:

1. Assume the element is one of those from the basis step and prove the conclusion

2. Assume the element is one of those built during the recursive step, **and assume that the property holds for the elements used to build it**, and prove the conclusion.

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

Analogy: unit tests in programming

Basis Step

1. **To Show** $length(\ append([], 100)\ ) > length([])$	Because $[]$ is the only element defined in the basis step of L, we only need to prove that the property 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 *length*.
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 Show** to equivalent statements, using well-defined proof techniques, and applying definitions.

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

Analogy: unit tests in programming

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 true. We evaluate each side of the candidate inequality. Applying the recursive definition of *append*,

$$\begin{aligned} LHS &= length(\ append((n, l'), 100)\ ) = length(\ (n, append(l', 100))\ ) && \text{by the recursive definition of } append \\ &= 1 + length(\ append(l', 100)\ ) && \text{by the recursive definition of } length \\ &> 1 + length(l') && \text{by the induction hypothesis} \\ &= length((n, l')) && \text{by the recursive definition of } length \\ &= RHS \end{aligned}$$

For next time

- Read website carefully

http://cseweb.ucsd.edu/classes/fa20/cse20-a/

Pre class reading for next time: Example 1 Section 5.1 p316