CS311H: Discrete Mathematics

Recursive Definitions and Structural Induction

Instructor: İsıl Dillig

Example

Let \( f_n \) denote the \( n \)'th element of the Fibonacci sequence

Prove: For \( n \geq 3 \), \( f_n > \alpha^{n-2} \) where \( \alpha = \frac{1 + \sqrt{5}}{2} \)

Proof is by strong induction on \( n \) with two base cases

Intuition 1: Definition of \( f_n \) has two base cases

Intuition 2: Recursive step uses \( f_{n-1}, f_{n-2} \) ⇒ strong induction

Base case 1 (\( n=3 \)): \( f_3 = 2 \), and \( \alpha < 2 \), thus \( f_3 > \alpha \)

Base case 2 (\( n=4 \)): \( f_4 = 3 \) and \( \alpha^2 = \frac{(3 + \sqrt{5})}{2} < 3 \)

Example, cont.

\[ \alpha^{k-1} = \alpha^{k-2} + \alpha^{k-3} \]

By recursive definition, we know \( f_{k+1} = f_k + f_{k-1} \)

Furthermore, by inductive hypothesis:

\[ f_k > \alpha^{k-2}, \quad f_{k-1} > \alpha^{k-3} \]

Therefore, \( f_{k+1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1} \)

Recursively Defined Sets and Structures

We can also define sets and other data structures recursively

Example: Consider the set \( S \) defined as:

\[ 3 \in S \]

If \( x \in S \) and \( y \in S \), then \( x + y \in S \)

What is the set \( S \) defined as above?

Announcements and Review

- Homework 5 due next lecture
- Review: Recursive definitions have base case and inductive step
- Typically use inductive proofs to prove properties about recursively defined sequences, functions, etc.
More Examples

- Give a recursive definition of the set $E$ of all even integers:
  - Base case:
  - Recursive step:

- Give a recursive definition of $\mathbb{N}$, the set of all natural numbers:
  - Base case:
  - Recursive step:

Strings and Alphabets

- Recursive definitions play important role in study of strings
- Strings are defined over an alphabet $\Sigma$
  - Example: $\Sigma_1 = \{a, b\}$
  - Example: $\Sigma_2 = \{0\}$
- Examples of strings over $\Sigma_1$: $a$, $b$, $aa$, $ba$, $bb$, $\ldots$
- Set of all strings formed from $\Sigma$ forms language called $\Sigma^*$
  - $\Sigma^* = \{\epsilon, 0, 00, 000, \ldots\}$

Recursive Definition of Strings

- The language $\Sigma^*$ has natural recursive definition:
  - Base case: $\epsilon \in \Sigma^*$ (empty string)
  - Recursive step: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$
  - Since $\epsilon$ is the empty string, $\epsilon s = s$
  - Consider the alphabet $\Sigma = \{0, 1\}$
  - How is the string “1” formed according to this definition?
  - How is “10” formed?

Recursive Definitions of String Operations

- Many operations on strings can be defined recursively.
  - Consider function $l(w)$ which yields length of string $w$
  - Example: Give recursive definition of $l(w)$
    - Base case:
    - Recursive step:

Another Example

- The reverse of a string $s$ is $s$ written backwards.
  - Example: Reverse of “abc” is “bca”
  - Give a recursive definition of the $\text{reverse}(s)$ operation
    - Base case:
    - Recursive step:

Palindromes

- A palindrome is a string that reads the same forwards and backwards
  - Examples: “mom”, “dad”, “abba”, “Madam I’m Adam”, $\ldots$
  - Give a recursive definition of the set $P$ of all palindromes over the alphabet $\Sigma = \{a, b\}$
    - Base cases:
    - Recursive step:
Structural Induction

- **Question**: How do we prove universally quantified properties about recursively-defined data (e.g., sets, strings)?
- **Structural induction** is a technique that allows us to apply induction on recursive definitions even if there is no integer
- Structural induction is also no more powerful than regular induction, but can make proofs much easier

**Example 1**

- Consider the following recursively defined set \( S \):
  1. \( a \in S \)
  2. If \( x \in S \), then \((x) \in S\)
- Prove by structural induction that every element in \( S \) contains an equal number of right and left parentheses.
  - **Base case**: \( a \) has 0 left and 0 right parentheses
  - **Inductive step**: By the inductive hypothesis, \( x \) has equal number, say \( n \), of right and left parentheses.
  - Thus, \((x)\) has \( n + 1 \) left and \( n + 1 \) right parentheses.

**Example 2**

- Consider the set \( S \) defined recursively as follows:
  - **Base case**: \( 3 \in S \)
  - **Recursive step**: If \( x \in S \) and \( y \in S \), then \( x + y \in S \)
- Prove \( S \) is set of all positive integers that are multiples of 3

**Proof, Part I**

Consider the set \( S \) defined recursively as follows: \( 3 \in S \) and if \( x \in S \) and \( y \in S \), then \( x + y \in S \)

**Proof, Part II**

- }
- }
- }
- }
- }
- }
- }
- }
- }
Proving Correctness of Reverse

- Earlier, we defined a \( \text{reverse}(w) \) function for length of strings:
  - **Base case:** \( \text{reverse}(\epsilon) = \epsilon \)
  - **Recursive step:** \( \text{reverse}(wx) = x \cdot \text{reverse}(w) \) where \( w \in \Sigma^* \) and \( x \in \Sigma \)
- Prove \( \forall x, y \in \Sigma^*. \text{reverse(xy)} = \text{reverse}(y) \cdot \text{reverse}(x) \)
- Let \( P(y) \) be the property \( \forall x \in \Sigma^*. \text{reverse(xy)} = \text{reverse}(y) \cdot \text{reverse}(x) \)
- We’ll prove by structural induction that \( \forall y \in \Sigma^*. P(y) \) holds, which is the desired property

Proof of Correctness of Reverse, cont.

- \( P(y) : \forall x \in \Sigma^*. \text{reverse(xy)} = \text{reverse}(y) \cdot \text{reverse}(x) \)

One More Reverse Example

- **Inductive step:** \( s = wx \) where \( w \in \Sigma^*, x \in \Sigma \)
  - Want to show:
  - Using previously shown lemma, \( \text{reverse}(\text{reverse}(wx)) = \text{reverse}(x) \cdot \text{reverse}(w) \)
  - Again, using previous lemma, \( \text{reverse}(\text{reverse}(x) \cdot \text{reverse}(w)) = \)
    - By inductive hypothesis, \( \text{reverse}(\text{reverse}(w)) = \)
    - Also by \( \text{IH} \), \( \text{reverse}(\text{reverse}(x)) = \)
    - Thus, \( \text{reverse}(\text{reverse}(w)) \cdot \text{reverse}(\text{reverse}(x)) = \text{wx} \)

Structural vs. Strong Induction

- Structural induction may look different from other forms of induction, but it is an implicit form of **strong induction**
- **Intuition:** We can define an integer \( k \) that represents how many times we need to use the recursive step in the definition
  - For base case, \( k = 0 \); if we use recursive step once, \( k = 1 \) etc.
General Induction and Well-Ordered Sets

- Inductive proofs can be used for any well-ordered set:
  1. have a least element
  2. total order between elements: either \( a \leq b \) or \( b \leq a \)
- Can use induction to prove properties of any well-ordered set:
  - Base case: Prove property about least element in set
  - Inductive step: To prove \( P(e) \), assume \( P(e') \) for all \( e' < e \)

Ordered Pairs of Natural Numbers

- Consider the set \( \mathbb{N} \times \mathbb{N} \), pairs of non-negative integers
- Let’s define the following order \( \leq \) on this set:
  \[
  (x_1, y_1) \leq (x_2, y_2) \text{ if } \begin{cases} 
  x_1 < x_2 \\
  x_1 = x_2 \land y_1 \leq y_2 
  \end{cases}
  \]
- This is an example of lexicographic order, which is a kind of total order; hence \( (\mathbb{N} \times \mathbb{N}, \leq) \) is a well-ordered set
- Question: What is the least element of this set?

Generalized Induction Example

- Suppose that \( a_{m,n} \) is defined recursively for \((m,n) \in \mathbb{N} \times \mathbb{N} \):
  \[
  a_{0,0} = 0 \\
  a_{m,n} = \begin{cases} 
  a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0 \\
  a_{m,n-1} + n & \text{if } n > 0 
  \end{cases}
  \]
- Show that \( a_{m,n} = m + n(n+1)/2 \)
- Proof is by induction on \((m,n)\) where \((m,n) \in (\mathbb{N} \times \mathbb{N}, \leq)\)

Example, cont.

Show \( a_{m,n} = m + n(n+1)/2 \) for:

\[
\begin{align*}
  a_{0,0} &= 0 \\
  a_{m,n} &= \begin{cases} 
  a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0 \\
  a_{m,n-1} + n & \text{if } n > 0 
  \end{cases}
\end{align*}
\]
Example, cont.

Show \(a_{m,n} = m + n(n+1)/2\) for:

\[
\begin{align*}
    a_{0,0} &= 0 \\
    a_{m,n} &= \begin{cases} 
        a_{m-1,n} + 1 & \text{if } n = 0 \text{ and } m > 0 \\
        a_{m,n-1} + n & \text{if } n > 0
    \end{cases}
\end{align*}
\]