Structural Induction

Jason Filippou

CMSC250 @ UMCP

07-05-2016

Outline

1. Recursively defined structures

2. Proofs
   - Binary Trees
   - Sets

Recursively defined structures

Recursively defined structures

- Many structures in Computer Science are *recursively defined*, i.e *parts of them exhibit the same characteristics and have the same properties as the whole!*
- They are also "well-ordered", in the sense that they exhibit a "well-founded partial order", like the order $\leq$ of $\mathbb{Z}$ or $\subseteq$ for sets.

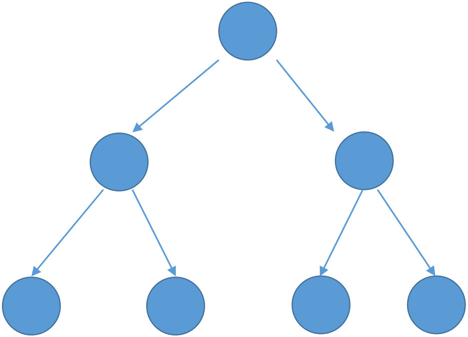
Structural induction as a proof methodology

- **Structural induction** is a proof methodology similar to mathematical induction, only instead of working in the domain of positive integers ($\mathbb{N}$) it works in the domain of such **recursively defined structures**!
- It is terrifically useful for proving *properties* of such structures.
- Its structure is sometimes "looser" than that of mathematical induction.

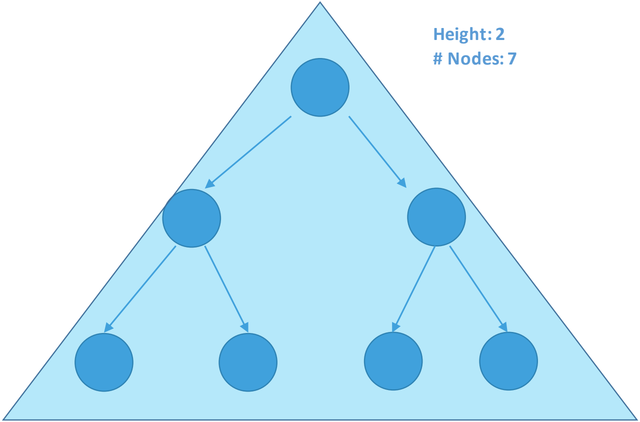
Proofs

Binary Trees

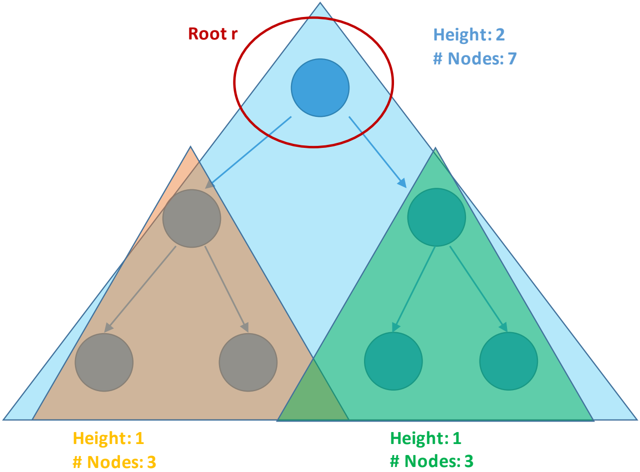
Pictorially



Pictorially



Pictorially



A recursive definition and statement on binary trees

Definition (Non-empty binary tree)

A **non-empty** binary tree T is either:

- **Base case:** A root node r with *no pointers*, or
- **Recursive (or inductive) step:** A root node r pointing to 2 *non-empty binary trees* T_L and T_R

A recursive definition and statement on binary trees

Definition (Non-empty binary tree)

A **non-empty** binary tree T is either:

- **Base case:** A root node r with *no pointers*, or
- **Recursive (or inductive) step:** A root node r pointing to 2 *non-empty binary trees* T_L and T_R

Claim: $|V| = |E| + 1$

The number of vertices ($|V|$) of a *non-empty binary tree* T is the number of its edges ($|E|$) plus one.

First structurally inductive proof

Proof (via structural induction on non-empty binary trees).

Let T be a non-empty binary tree and P the proposition we want to hold..

1. **Inductive Base**: If T consists of a single root node r (base case for a non-empty binary tree), then $|V| = 1$ and $|E| = 0$, so $P(r)$ holds.
2. **Inductive Hypothesis**: In the recursive part of the definition for a non-empty binary tree, T may consist of a root node r pointing to 1 or 2 non-empty binary trees T_L and T_R. Without loss of generality, we can assume that both T_L and T_R are defined, and we assume $P(T_L)$ and $P(T_R)$.
3. **Inductive Step**: We prove now that $P(T)$ must hold. Denote by V_L, E_L, V_R, E_R the vertex and edge sets of the left and right subtrees respectively. We obtain:

$$\begin{aligned}
|V| &= |V_L| + |V_R| + 1 && \textit{(By definition of non-empty binary trees)} \\
&= (|E_L| + 1) + (|E_R| + 1) + 1 && \textit{(By the Inductive Hypothesis)} \\
&= (|E_L| + |E_R| + 2) + 1 && \textit{(By grouping terms)} \\
&= |E| + 1 && \textit{(By definition of non-empty binary trees)}
\end{aligned}$$

So $P(T)$ holds. □

Here's one for you!

Definition (Height of a non-empty binary tree)

The height $h(T)$ of a non-empty binary tree T is defined as follows:

- **(Base case:)** If T is a single root node r, $h(r) = 0$.
- **(Recursive step:)** If T is a root node connected to two "sub-trees" T_L and T_R, $h(T) = max\{h(T_R), h(T_L)\} + 1$

Theorem ($m(T)$ as a function of $h(T)$)

A non-empty binary tree T of height $h(T)$ has ***at most*** $2^{h(T)+1} - 1$ *nodes.*

General Structure of structurally inductive proofs on trees

1. **Prove** $P(\cdot)$ for the base-case of the tree.

General Structure of structurally inductive proofs on trees

1. **Prove** $P(\cdot)$ for the base-case of the tree.
   - This can either be an empty tree, or a trivial "root" node, say r. That is, you will **prove** something like $P(\texttt{null})$ or $P(r)$.

General Structure of structurally inductive proofs on trees

1. **Prove** $P(\cdot)$ for the base-case of the tree.
    - This can either be an empty tree, or a trivial "root" node, say r. That is, you will **prove** something like $P(\texttt{null})$ or $P(r)$.
    - As always, **prove explicitly!**

General Structure of structurally inductive proofs on trees

1. **Prove** $P(\cdot)$ for the base-case of the tree.
   - This can either be an empty tree, or a trivial "root" node, say r. That is, you will **prove** something like $P(\texttt{null})$ or $P(r)$.
   - As always, **prove explicitly!**
2. **Assume** the inductive hypothesis for an arbitrary tree T, i.e assume $P(T)$.
   - Valid to do so, since at least for the trivial case we have explicit proof!

General Structure of structurally inductive proofs on trees

1. **Prove** $P(\cdot)$ for the base-case of the tree.
   - This can either be an empty tree, or a trivial "root" node, say r. That is, you will **prove** something like $P(\texttt{null})$ or $P(r)$.
   - As always, **prove explicitly!**
2. **Assume** the inductive hypothesis for an arbitrary tree T, i.e assume $P(T)$.
   - Valid to do so, since at least for the trivial case we have explicit proof!
3. Use the inductive / recursive part of the tree's definition to build a new tree, say T', from existing (sub-)trees T_i, and **prove** $P(T')$!

General Structure of structurally inductive proofs on trees

1. **Prove** $P(\cdot)$ for the base-case of the tree.
    - This can either be an empty tree, or a trivial "root" node, say r. That is, you will **prove** something like $P(\texttt{null})$ or $P(r)$.
    - As always, **prove explicitly!**
2. **Assume** the inductive hypothesis for an arbitrary tree T, i.e assume $P(T)$.
    - Valid to do so, since at least for the trivial case we have explicit proof!
3. Use the inductive / recursive part of the tree's definition to build a new tree, say T', from existing (sub-)trees T_i, and **prove** $P(T')$!
    - Use the Inductive Hypothesis on the T_i!

Sets

Recursive definitions of sets

- Sets can be defined **recursively**!
- Our goal is to find a “flat” definition of them (a *“closed-form” description*), much in the same way we did with recursive sequences and strong induction.
- Consider the following:

Recursive definitions of sets

- Sets can be defined **recursively**!
- Our goal is to find a "flat" definition of them (a *"closed-form" description*), much in the same way we did with recursive sequences and strong induction.
- Consider the following:
  1. S_1 is such that $3 \in S_1$ **(base case)** and if $x, y \in S_1$, then $x + y \in S_1$ (**recursive step**).

Recursive definitions of sets

- Sets can be defined **recursively**!
- Our goal is to find a "flat" definition of them (a *"closed-form" description*), much in the same way we did with recursive sequences and strong induction.
- Consider the following:
  1. S_1 is such that $3 \in S_1$ **(base case)** and if $x, y \in S_1$, then $x + y \in S_1$ (**recursive step**).
  2. S_2 is such that $2 \in S_2$ and if $x \in S_2$, then $x^2 \in S_2$.

Recursive definitions of sets

- Sets can be defined **recursively**!
- Our goal is to find a "flat" definition of them (a *"closed-form" description*), much in the same way we did with recursive sequences and strong induction.
- Consider the following:
  1. S_1 is such that $3 \in S_1$ **(base case)** and if $x, y \in S_1$, then $x + y \in S_1$ (**recursive step**).
  2. S_2 is such that $2 \in S_2$ and if $x \in S_2$, then $x^2 \in S_2$.
  3. S_3 is such that $0 \in S_3$ and if $y \in S_3$, then $y + 1 \in S_3$.

Recursive definitions of sets

- Sets can be defined **recursively**!
- Our goal is to find a "flat" definition of them (a *"closed-form" description*), much in the same way we did with recursive sequences and strong induction.
- Consider the following:
  1. S_1 is such that $3 \in S_1$ **(base case)** and if $x, y \in S_1$, then $x + y \in S_1$ (**recursive step**).
  2. S_2 is such that $2 \in S_2$ and if $x \in S_2$, then $x^2 \in S_2$.
  3. S_3 is such that $0 \in S_3$ and if $y \in S_3$, then $y + 1 \in S_3$.

  - Vote ($>$ 1 possible)! $2 \in$ S_1 S_2 S_3

Recursive definitions of sets

- Sets can be defined **recursively**!
- Our goal is to find a "flat" definition of them (a *"closed-form" description*), much in the same way we did with recursive sequences and strong induction.
- Consider the following:
  1. S_1 is such that $3 \in S_1$ **(base case)** and if $x, y \in S_1$, then $x + y \in S_1$ (**recursive step**).
  2. S_2 is such that $2 \in S_2$ and if $x \in S_2$, then $x^2 \in S_2$.
  3. S_3 is such that $0 \in S_3$ and if $y \in S_3$, then $y + 1 \in S_3$.

  - Vote ($>$ 1 possible)! $2 \in$ S_1 S_2 S_3
  - $16 \in$ S_1 S_2 S_3

Recursive definitions of sets

- Sets can be defined **recursively**!
- Our goal is to find a "flat" definition of them (a *"closed-form" description*), much in the same way we did with recursive sequences and strong induction.
- Consider the following:
  1. S_1 is such that $3 \in S_1$ **(base case)** and if $x, y \in S_1$, then $x + y \in S_1$ (**recursive step**).
  2. S_2 is such that $2 \in S_2$ and if $x \in S_2$, then $x^2 \in S_2$.
  3. S_3 is such that $0 \in S_3$ and if $y \in S_3$, then $y + 1 \in S_3$.

  - Vote ($>$ 1 possible)! $2 \in$ S_1 S_2 S_3
  - $16 \in$ S_1 S_2 S_3
  - $21 \in$ S_1 S_2 S_3

Practice!

4. S_4 is such that $1 \in S_4$ and if $x, y \in S_4$, then $x(-y) \in S_4$.
5. S_5 is such that $\emptyset \in S_5$ and $a \in S_5 \Rightarrow \{a\} \in S_5$.
6. S_6 is such that $\emptyset \in S_6$ and $a, b \in S_6 \Rightarrow a \cup b \in S_6$.
7. S_7 is such that $\{0\} \in S_7$ and $(x \in S_7) \wedge (i \in \mathbb{N}^*) \Rightarrow x \cap \{i\} \in S_7$.
8. S_8 is such that $i \in \mathbb{N}^* \Rightarrow \{i\} \cap \{i+1\} \in S_8$.

Practice!

4. S_4 is such that $1 \in S_4$ and if $x, y \in S_4$, then $x(-y) \in S_4$.
5. S_5 is such that $\emptyset \in S_5$ and $a \in S_5 \Rightarrow \{a\} \in S_5$.
6. S_6 is such that $\emptyset \in S_6$ and $a, b \in S_6 \Rightarrow a \cup b \in S_6$.
7. S_7 is such that $\{0\} \in S_7$ and $(x \in S_7) \wedge (i \in \mathbb{N}^*) \Rightarrow x \cap \{i\} \in S_7$.
8. S_8 is such that $i \in \mathbb{N}^* \Rightarrow \{i\} \cap \{i+1\} \in S_8$.

- $|S_4| =$

0 1 2 $+\infty$

Practice!

4. S_4 is such that $1 \in S_4$ and if $x, y \in S_4$, then $x(-y) \in S_4$.
5. S_5 is such that $\emptyset \in S_5$ and $a \in S_5 \Rightarrow \{a\} \in S_5$.
6. S_6 is such that $\emptyset \in S_6$ and $a, b \in S_6 \Rightarrow a \cup b \in S_6$.
7. S_7 is such that $\{0\} \in S_7$ and $(x \in S_7) \wedge (i \in \mathbb{N}^*) \Rightarrow x \cap \{i\} \in S_7$.
8. S_8 is such that $i \in \mathbb{N}^* \Rightarrow \{i\} \cap \{i+1\} \in S_8$.

- $|S_4| =$ 0 1 2 $+\infty$
- $|S_5| =$ 0 1 2 $+\infty$

Practice!

4. S_4 is such that $1 \in S_4$ and if $x, y \in S_4$, then $x(-y) \in S_4$.
5. S_5 is such that $\emptyset \in S_5$ and $a \in S_5 \Rightarrow \{a\} \in S_5$.
6. S_6 is such that $\emptyset \in S_6$ and $a, b \in S_6 \Rightarrow a \cup b \in S_6$.
7. S_7 is such that $\{0\} \in S_7$ and $(x \in S_7) \wedge (i \in \mathbb{N}^*) \Rightarrow x \cap \{i\} \in S_7$.
8. S_8 is such that $i \in \mathbb{N}^* \Rightarrow \{i\} \cap \{i+1\} \in S_8$.

- $|S_4| =$ 0 1 2 $+\infty$
- $|S_5| =$ 0 1 2 $+\infty$
- $|S_6| =$ 0 1 2 $+\infty$

Practice!

4. S_4 is such that $1 \in S_4$ and if $x, y \in S_4$, then $x(-y) \in S_4$.
5. S_5 is such that $\emptyset \in S_5$ and $a \in S_5 \Rightarrow \{a\} \in S_5$.
6. S_6 is such that $\emptyset \in S_6$ and $a, b \in S_6 \Rightarrow a \cup b \in S_6$.
7. S_7 is such that $\{0\} \in S_7$ and $(x \in S_7) \wedge (i \in \mathbb{N}^*) \Rightarrow x \cap \{i\} \in S_7$.
8. S_8 is such that $i \in \mathbb{N}^* \Rightarrow \{i\} \cap \{i+1\} \in S_8$.

- $|S_4| =$ 0 1 2 $+\infty$
- $|S_5| =$ 0 1 2 $+\infty$
- $|S_6| =$ 0 1 2 $+\infty$
- $|S_7| =$ 0 1 2 $+\infty$

Practice!

4. S_4 is such that $1 \in S_4$ and if $x, y \in S_4$, then $x(-y) \in S_4$.
5. S_5 is such that $\emptyset \in S_5$ and $a \in S_5 \Rightarrow \{a\} \in S_5$.
6. S_6 is such that $\emptyset \in S_6$ and $a, b \in S_6 \Rightarrow a \cup b \in S_6$.
7. S_7 is such that $\{0\} \in S_7$ and $(x \in S_7) \wedge (i \in \mathbb{N}^*) \Rightarrow x \cap \{i\} \in S_7$.
8. S_8 is such that $i \in \mathbb{N}^* \Rightarrow \{i\} \cap \{i+1\} \in S_8$.

- $|S_4| =$ 0 1 2 $+\infty$
- $|S_5| =$ 0 1 2 $+\infty$
- $|S_6| =$ 0 1 2 $+\infty$
- $|S_7| =$ 0 1 2 $+\infty$
- $|S_8| =$ 0 1 2 $+\infty$

Our first structurally inductive proof on sets

A proposition on a recursively defined set

Let S be a set defined as follows:

- **Base case**: $4 \in S$
- **Recursive / Inductive step**: If $x \in S$, then $x^2 \in S$.

Then, prove that $\forall x \in S$, x is even.

Our first structurally inductive proof on sets

A proposition on a recursively defined set

Let S be a set defined as follows:

- **Base case**: $4 \in S$
- **Recursive / Inductive step**: If $x \in S$, then $x^2 \in S$.

Then, prove that $\forall x \in S$, x is even.

Proof

Proof (via structural induction on S).

Let P be the proposition we want to prove. We proceed inductively:

- **Inductive base**: In the base case of the definition of S, we have that $4 \in S$. Since 4 is an even number, $P(\{4\})$ holds.
- **Inductive hypothesis**: The inductive definition of S successively builds sets S' from previous "versions" of S. We assume that $\exists S' : |S'| \geq 1$ and $P(S')$.
- **Inductive step**: We will prove that $P(S' \cup \{x^2 | x \in S'\})$ holds. Let x_0 be an arbitrarily selected element of S'. From the inductive hypothesis, we know that x_0 is even. From a known theorem, we know that x_0^2 is even. But this means that $P(\{x_0\})$, and since x_0 was arbitrarily selected within S', $P(S' \cup \{x^2 | x \in S'\})$ holds.

□

A proof on the Cartesian Plane

An inequality proof

Let S be the subset of $\mathbb{Z} \times \mathbb{Z} = \mathbb{Z}^2$ defined as follows:

- **Base case**: $(0, 0) \in S$
- **Recursive step**:
  $(a, b) \in S \Rightarrow (((a, b+1) \in S) \wedge ((a+1, b+1) \in S) \wedge (a+2, b+1) \in S)$.
  Prove that $\forall (a, b) \in S, a \leq 2b$.

- *Suggestion:* To make sure you understand the exercise, first list 5 elements in S.

Cartesian plane exercise, proof

In class!

Set equality and structural induction

The set of positive multiples of 3

Let the set S be such that:

- **(Base case:)** $3 \in S$
- **(Recursive step:)** $(x \in S) \wedge (y \in S) \Rightarrow (x + y) \in S$

Then, $S = \{3n, \ \forall n \in \mathbb{N}^*\}$.

Set equality and structural induction

The set of positive multiples of 3

Let the set S be such that:

- **(Base case:)** $3 \in S$
- **(Recursive step:)** $(x \in S) \wedge (y \in S) \Rightarrow (x + y) \in S$

Then, $S = \{3n, \ \forall n \in \mathbb{N}^*\}$.

- What do I need to do in order to prove this statement?

Set equality and structural induction

The set of positive multiples of 3

Let the set S be such that:

- **(Base case:)** $3 \in S$
- **(Recursive step:)** $(x \in S) \wedge (y \in S) \Rightarrow (x + y) \in S$

Then, $S = \{3n, \ \forall n \in \mathbb{N}^*\}$.

- What do I need to do in order to prove this statement?
- $A \subseteq S$

Contradiction | Cases | Mathematical induction | Structural Induction

Set equality and structural induction

The set of positive multiples of 3

Let the set S be such that:

- **(Base case:)** $3 \in S$
- **(Recursive step:)** $(x \in S) \wedge (y \in S) \Rightarrow (x + y) \in S$

Then, $S = \{3n, \ \forall n \in \mathbb{N}^*\}$.

- What do I need to do in order to prove this statement?
- $A \subseteq S$

  Contradiction | Cases | Mathematical induction | Structural Induction

- $S \subseteq A$

  Contradiction | Cases | Mathematical induction | Structural Induction

Proof

Proof (via weak induction on n and structural induction on S!)

Let $A = \{3n,\ \forall n \in \mathbb{N}^*\}$. To prove $S = A$, we need to prove that $S \subseteq A$ and $A \subseteq S$.

- First, we prove that $A \subseteq S$. The proof is via weak induction on n. Let r be a generic particular for $\mathbb{N}^*$ and $P(n)$ be the statement we want to prove.[a] We proceed inductively:
  1. **Inductive base**: For $r = 1$, $3r = 3 \cdot 1 \in S$. Therefore, $P(1)$ holds.
  2. **Inductive hypothesis**: We assume that for some value of $r \geq 1$, $P(r)$ holds, i.e $3r \in S$
  3. **Inductive step**: We want to prove $P(r+1)$, that is, $3(r+1) \in S$. We know that $3r \in S$ and $3 \in S$ by the inductive base and hypothesis. By the definition of S, this means that $3r + 3 \in S \overset{(Algebra)}{\iff} 3(r+1) \in S$. So, $P(r+1)$ holds.

  Since $r \in \mathbb{N}^*$ was arbitrarily chosen, the result holds for all $n \in \mathbb{N}^*$.

[a] We can do this, since A is parameterized by n.

Proof (continued)

Proof: $S \subseteq A$ part.

- We now prove that $S \subseteq A$.
  1. **Inductive base**: By the base case of the definition of S, we have that $3 \in S$. Since $3 = 3 \cdot 1$ and $1 \in \mathbb{N}^*$, we have that $3 \in A$. So $P(3)$.
  2. **Inductive hypothesis**: Assume that $x, y \in S$ are also contained by A, i.e $x, y \in A$. So $P(x)$, $P(y)$.
  3. **Inductive step**: We must show that $P(x+y)$, i.e $x + y \in A$, because if we do show this, we will have covered the recursive step of A's definition. From the inductive hypothesis, we have that $x, y \in A \Rightarrow \exists i, j \in \mathbb{N}^* : x = 3i,\ y = 3j$. Therefore, $x + y = 3\underbrace{(i+j)}_{z \in \mathbb{N}} \Rightarrow x + y \in A$. Therefore, $P(x+y)$.

□

Recursively defined languages!

Definition

A recursively defined language Let $\Sigma = \{a, b\}$ be an alphabet. We define the language $\mathcal{L}$ as follows:

- **(Base case:)** $\epsilon \in \mathcal{L}$
- **(Recursive step:)** If $x \in \mathcal{L}$, $axa \in \mathcal{L}$ and $bxb \in \mathcal{L}$.[a]

---

[a] $\sigma_1\sigma_2$ is the **concatenation** of σ_1 and σ_2. Concatenation can be applied to n strings, $n = 1, 2, 3, \ldots$. σ^n is the concatenation of σ $n-$ many times.

Recursively defined languages!

Definition

A recursively defined language Let $\Sigma = \{a, b\}$ be an alphabet. We define the language $\mathcal{L}$ as follows:

- **(Base case:)** $\epsilon \in \mathcal{L}$
- **(Recursive step:)** If $x \in \mathcal{L}$, $axa \in \mathcal{L}$ and $bxb \in \mathcal{L}$.[a]

[a] $\sigma_1\sigma_2$ is the **concatenation** of σ_1 and σ_2. Concatenation can be applied to n strings, $n = 1, 2, 3, \ldots$. σ^n is the concatenation of σ $n-$ many times.

Claim

$\forall\sigma \in \mathcal{L}, |\sigma|$ is even.[a]

[a] $|\sigma|$ is the number of characters in σ.

General structure on inductive proofs on sets

1. In the inductive base, **prove** $P(\cdot)$ for all the base elements of the recursively defined set S.

General structure on inductive proofs on sets

1. In the inductive base, **prove** $P(\cdot)$ for all the base elements of the recursively defined set S.
   - This gives us that $P(\cdot)$ holds for sets up to some cardinality n_0.

General structure on inductive proofs on sets

1. In the inductive base, **prove** $P(\cdot)$ for all the base elements of the recursively defined set S.
   - This gives us that $P(\cdot)$ holds for sets up to some cardinality n_0.
2. In the inductive hypothesis, **assume** $\exists S : |S| \geq n_0$ and $P(S)$.

General structure on inductive proofs on sets

1. In the inductive base, **prove** $P(\cdot)$ for all the base elements of the recursively defined set S.
   - This gives us that $P(\cdot)$ holds for sets up to some cardinality n_0.
2. In the inductive hypothesis, **assume** $\exists S : |S| \geq n_0$ and $P(S)$.
   - Reminds us of weak mathematical induction?

General structure on inductive proofs on sets

1. In the inductive base, **prove** $P(\cdot)$ for all the base elements of the recursively defined set S.
    - This gives us that $P(\cdot)$ holds for sets up to some cardinality n_0.
2. In the inductive hypothesis, **assume** $\exists S : |S| \geq n_0$ and $P(S)$.
    - Reminds us of weak mathematical induction?
3. In the inductive step, use the recursive part of the definition of S to prove that the new set constructed (call it S') satisfies the proposition (so $P(S')$).
    - The inductive hypothesis will undoubtedly be used.