COMPSCI/SFWRENG 2FA3
Discrete Mathematics with Applications II
Winter 2020

2 Recursion and Induction

William M. Farmer

Department of Computing and Software
McMaster University

February 3, 2020

Admin — January 10

- In memory of Iman Aghabali and Mehdi Eshaghian and the other victims of UIA flight PS752 that crashed in Iran.
- Week 02 Exercises are posted on Avenue. The solutions will be posted at the end of next week.
- Assignment 1 and Extra Credit Assignment 1 will be posted at the end of next week.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Review

- What is a mathematical proof
- Purposes of mathematical proofs
- Styles of mathematical proofs
- Traditional proofs vs. formal proofs
- Methods of proof
- Proof terminology

Mathematical Proof (iClicker)

How much of the 1 Mathematical Proof lecture was a review?

A. 0%.
B. 25%.
C. 50%
D. 75%.
E. 100%

What is Recursion?

- Recursion is a method of defining a function or structure in terms of itself.
  - One of the most fundamental ideas of computing.
  - Can make specifications, descriptions, and programs easier to express, understand, and prove correct.
- A problem is solved by recursion as follows:
  1. The simplest instances of the problem are solved directly.
  2. Each other instance of the problem is solved by reducing the instance to simpler instances of the problem.
  3. As a result of 1 and 2, each instance can be solved by reducing the instance to simpler instances and then reducing these instances to simpler instances and continuing in this fashion until a simplest instance is reached, which has already been solved.
- Recursion employs a divide and conquer strategy.

What is Induction?

- Induction is a method of proof based on a inductive set, a well-order, or a well-founded relation.
  - Most important proof technique used in computing.
  - The proof method is specified by an induction principle.
  - Induction is especially useful for proving properties about recursively defined functions.
- Note: The terms "recursion" and "induction" are often used interchangeably — which is confusing and unfortunate!

Outline

1. Natural number recursion and induction.
2. Structural recursion and induction.
3. Orders.
4. Ordinal recursion and induction.
5. Well-founded recursion and induction.
6. Summary

1. Natural Number Recursion and Induction

Natural Number Recursion

- Let $(\mathbb{N}, <)$ be the natural numbers with the usual order.
- A function $f : I \to O$ is defined by natural number recursion as follows:
  - The definition of f has the form
    $$f(x) = E(f(a_0(x)), \ldots, f(a_n(x))).$$
  - Each $i \in I$ is assigned a complexity $c(i) \in \mathbb{N}$.
  - For all $i \in I$ and $m \in \mathbb{N}$ with $0 \leq m \leq n$,
    $$c(a_m(i)) < c(i).$$
- This approach works since $(\mathbb{N}, <)$ is Noetherian, i.e., every descending sequence
  $$\cdots < n_2 < n_1 < n_0.$$
  of natural numbers is finite.
- A recursive definition of a function is nonsensical if, for some $i \in I$ and $m \in \mathbb{N}$, $c(a_m(i)) \geq c(i)$.

Example: Recursively Defined Function

- Let $h : \mathbb{N} \times \mathbb{N} \times (\mathbb{N} \rightarrow \mathbb{R}) \rightarrow \mathbb{R}$ be defined as:

  $$h(m, n, f) = \begin{cases} 0 & \text{if } m > n \\ f(n) + h(m, n-1, f) & \text{if } m \leq n \end{cases}$$

  Notice that

  $$h(m, n, f) = \sum_{i=m}^{n} f(i).$$

- h is defined by natural number recursion using the following complexity function:

  $$c(m, n, f) = \begin{cases} 0 & \text{if } m > n \\ n - m + 1 & \text{if } m \leq n \end{cases}$$

  We must check that, for all $m, n \in \mathbb{N}$, with $m \leq n$,

  $$c(m, n-1, f) < c(m, n, f).$$

Natural Number Induction

- Weak induction is:

$$(P(0) \wedge \forall x \in \mathbb{N} \,.\, (P(x) \Rightarrow P(x+1))) \Rightarrow \forall x \in \mathbb{N} \,.\, P(x)$$

  holds for every property P of $\mathbb{N}$. This induction principle is also called mathematical induction.

- Strong induction is:

$$\forall x \in \mathbb{N} \,.\, (\forall y \in \mathbb{N} \,.\, (y < x \Rightarrow P(y)) \Rightarrow P(x))$$
$$\Rightarrow \forall x \in \mathbb{N} \,.\, P(x)$$

  holds for every property P of $\mathbb{N}$. This induction principle is also called complete induction and course-of-values induction.

- Theorem. The following are equivalent:
  1. Weak induction.
  2. Strong induction.

Admin — January 14

- M&Ms.
- Exercises and assignments.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Review

- Natural number recursion.
- Weak induction.
- Strong induction.

Weak Induction vs. Strong Induction

- Weak induction:

  $$(P(0) \wedge \forall x \in \mathbb{N} \,.\, (P(x) \Rightarrow P(x+1))) \Rightarrow \forall x \in \mathbb{N} \,.\, P(x).$$

  Form of Proof:

  1. Base case: $x = 0$. Show $P(0)$.
  2. Induction step: $x \geq 0$. Assume $P(x)$. Show $P(x+1)$.

- Strong induction:

  $$\forall x \in \mathbb{N} \,.\, (\forall y \in \mathbb{N} \,.\, (y < x \Rightarrow P(y)) \Rightarrow P(x)) \\ \Rightarrow \forall x \in \mathbb{N} \,.\, P(x).$$

  Form of Proof:

  1. Base case: $x = 0$. (Assume nothing.) Show $P(0)$.
  2. Induction step: $x > 0$. Assume $P(y)$ for all $y < x$. Show $P(x)$.

- Strong induction is called "strong" because it provides a stronger induction hypothesis than weak induction.

Example: Weak Induction

Theorem. $\forall\, n \in \mathbb{N} \,.\, \sum_{i=0}^{n-1} 2^i = 2^n - 1$.

Proof. Let $P(n)$ be $\sum_{i=0}^{n-1} 2^i = 2^n - 1$. We will prove $P(n)$ for all $n \in \mathbb{N}$ by weak induction.

Base case: $n = 0$. We must show $P(0)$.

$$\sum_{i=0}^{0-1} 2^i = \sum_{i=0}^{-1} 2^i = 0 = 1 - 1 = 2^0 - 1$$

So $P(0)$ holds.

Induction step: $n \geq 0$. Assume $P(n)$. We must show $P(n+1)$.

$$\begin{aligned}
\sum_{i=0}^{(n+1)-1} 2^i &= \sum_{i=0}^{n} 2^i && \langle \text{arithmetic} \rangle \\
&= 2^n + \sum_{i=0}^{n-1} 2^i && \langle \text{definition of } \textstyle\sum_{i=m}^{n} f(i) \rangle \\
&= 2^n + 2^n - 1 && \langle \text{induction hypothesis} \rangle \\
&= 2 * 2^n - 1 && \langle \text{arithmetic} \rangle \\
&= 2^{n+1} - 1 && \langle \text{arithmetic} \rangle
\end{aligned}$$

So $P(n+1)$ holds. Thus, the theorem holds by weak induction.

Example: Strong Induction

Theorem. If $n \in \mathbb{N}$ with $n \geq 2$, then n is a product of prime numbers.

Proof. Let $P(n)$ be $n = p_0 * \cdots * p_m$ where $m \geq 0$ and $p_0, \ldots, p_m$ are prime numbers. We will prove $P(n)$ for all $n \geq 2$ by strong induction.

Base case: $n = 2$. We must show $P(2)$. But $P(2)$ holds since 2 is a prime number.

Induction step: $n > 2$. Assume $P(2), \ldots, P(n-1)$ hold. We must show $P(n)$. If n is a prime number, then $P(n)$ holds. Otherwise, $n = x * y$ with $2 \leq x, y \leq n-1$. By the induction hypothesis, $P(x)$ and $P(y)$ hold, so $x = p_0 * \cdots * p_i$ and $y = q_0 * \cdots * q_j$ where $p_0, \ldots, p_i, q_0, \ldots, q_j$ are prime numbers. Thus, $n = p_0 * \cdots * p_i * q_0 * \cdots * q_j$ and so $P(n)$ holds.

Therefore, the theorem holds by strong induction.

General Form of a Proof by Induction

A proof by induction should have the following components:

1. The definition of the relevant property P.
2. The theorem A of the form

   $$\forall x \in S \,.\, P(x)$$

   that is to be proved.
3. The induction principle I to be used in the proof.
4. Verification of the cases needed for induction principle I to be applied. (The cases include one or more base cases and one or more induction steps.)
5. A concluding statement that the theorem A has been proved by the induction principle I.

2. Structural Recursion and Induction

Inductive Sets [1/2]

- An inductive set (or inductive type) is a set S defined by a finite set of constructors (where $m_1, \ldots, m_n \geq 0$)

$$\begin{aligned} C_1 &: S^1_1 \times \cdots \times S^1_{m_1} \to S. \\ &\vdots \\ C_n &: S^n_1 \times \cdots \times S^n_{m_n} \to S. \end{aligned}$$

  such that each $a \in S$ can be constructed from the constructors in exactly one way.
  - That is, "no junk and no confusion".
- S is recursive if some of the sets S^i_j are S itself.
  - A recursive inductive set is well-defined iff it contains a member (called a base case) that is not constructed from other members of the set.
- The constructors $C_1, \ldots, C_n$ define a language whose expressions serve as literals for the members of S.

Inductive Sets [2/2]

- The definition of t induces a structural induction principle: A property P holds for all members of S provided for every constructor C_i

  if P holds for every $x_j \in S$ in $C_i(x_1, \ldots, x_{m_i})$, then P holds for $C_i(x_1, \ldots, x_{m_i})$.

  Less formally, a property P holds for all members of S provided:

  1. P holds for all members of S having minimal structure.
  2. P holds for a structural combination of members of S whenever it holds for the members themselves.

- A function f on S can be defined by structural recursion using pattern matching.
  - Each recursive application of f must be applied to at least one argument with reduced structure.
  - f is defined by several equations, one for each pattern.

Inductive Sets (iClicker)

How familiar are you with the definition of an inductive set (a.k.a., inductive type or algebraic type)?

A. Never saw it before.
B. Have seen it, but not understood it.
C. Understand it, but never used it.
D. Have used it occasionally.
E. Have routinely defined inductive sets using a programming language (e.g., Haskell).

Natural Numbers (iClicker)

How many constructors are needed to define the natural numbers as an inductive set?

A. 1.

B. $\boxed{\text{2.}}$

C. 3.

D. 4.

Example 1: Natural Numbers as an Inductive Set

- Nat is the inductive set representing the natural numbers defined by the following constructors:
  1. $0 : \mathsf{Nat}$ (i.e., $0 : \to \mathsf{Nat}$).
  2. $S : \mathsf{Nat} \to \mathsf{Nat}$.
- Nat is recursive.
- The members of Nat correspond to the expressions

  $$0, S(0), S(S(0)), \ldots$$

  which denote the natural numbers

  $$0, 1, 2, \ldots.$$

- The structural induction principle for Nat is:

  $$\begin{array}{l}(P(0) \wedge \forall\, x \in \mathsf{Nat} \,.\, (P(x) \Rightarrow P(S(x)))) \\ \quad \Rightarrow \forall\, x \in \mathsf{Nat} \,.\, P(x)\end{array}$$

  holds for every property P of Nat.
  This principle is weak induction!

Example 1: Functions Defined by Pattern Matching

- Addition ($+ : \mathsf{Nat} \times \mathsf{Nat} \rightarrow \mathsf{Nat}$) is defined by pattern matching as:
  1. $x + 0 = x$.
  2. $x + S(y) = S(x + y)$.
- Multiplication ($* : \mathsf{Nat} \times \mathsf{Nat} \rightarrow \mathsf{Nat}$) is defined by pattern matching as:
  1. $x * 0 = 0$.
  2. $x * S(y) = (x * y) + x$.
- The function $\mathsf{fib} : \mathsf{Nat} \rightarrow \mathsf{Nat}$ that maps n to the nth Fibonacci number is defined by pattern matching as:
  1. $\mathsf{fib}(0) = 0$.
  2. $\mathsf{fib}(S(0)) = S(0)$.
  3. $\mathsf{fib}(S(S(x))) = \mathsf{fib}(S(x)) + \mathsf{fib}(x)$.

Example 1: Structural Induction for Nat

Theorem. $\forall x \in \mathsf{Nat} \,.\, 0 + x = x$.

Proof. Let $P(x) \equiv 0 + x = x$. We will prove $P(x)$ for all $x \in \mathsf{Nat}$ by structural induction.

Base case: $x = 0$. We must show $P(0)$. $P(0) \equiv 0 + 0 = 0$ by the definition of P. $0 + 0 = 0$ is an instance of $x + 0 = x$. Hence $P(0)$ holds.

Induction step: $x = S(y)$. Assume $P(y)$. Show $P(S(y))$.

$$
\begin{array}{ll}
0 + S(y) = S(0 + y) & \langle \text{instance of } x + S(y) = S(x + y) \rangle \\
0 + y = y & \langle \text{induction hypothesis} \rangle \\
0 + S(y) = S(y) & \langle \text{equality reasoning using (1) and (2)} \rangle
\end{array}
$$

Hence $P(S(y))$ holds.

Thus, the theorem holds by structural induction.

Base Cases (iClicker)

A constructor $C : S_1 \times \cdots \times S_m \to S$ for an inductive set produces a base case if

A. $\boxed{\text{There are no } S_i \text{ (i.e., } C \text{ is 0-ary).}}$

B. $\boxed{\text{None of the } S_i \text{ are } S.}$

C. Some of the S_i are S.

D. All of the S_i are S.

Example 2: Binary Trees of Natural Numbers

- BinTree is the inductive set representing binary trees of natural numbers defined by the following constructors:

  1. $\mathsf{Leaf} : \mathsf{Nat} \to \mathsf{BinTree}$.
  2. $\mathsf{Branch} : \mathsf{BinTree} \times \mathsf{Nat} \times \mathsf{BinTree} \to \mathsf{BinTree}$.

- BinTree is recursive.
- The structural induction principle for BinTree is:

$$
\begin{array}{l}
(\forall\, n \in \mathsf{Nat}\,.\, P(\mathsf{Leaf}(n)) \wedge \\
\;\forall\, n \in \mathsf{Nat}\,.\, \forall\, t_1, t_2 \in \mathsf{BinTree}\,. \\
\qquad ((P(t_1) \wedge P(t_2)) \Rightarrow P(\mathsf{Branch}(t_1, n, t_2)))) \\
\quad \Rightarrow \forall\, t \in \mathsf{BinTree}\,.\, P(t)
\end{array}
$$

  holds for every property P of BinTree.

Example 2: Functions Defined by Pattern Matching

- The function $\mathsf{nodes} : \mathsf{BinTree} \to \mathsf{Nat}$ that maps a binary tree to the number of nodes in it is defined by pattern matching as:
  1. $\mathsf{nodes}(\mathsf{Leaf}(n)) = 1$.
  2. $\mathsf{nodes}(\mathsf{Branch}(t_1, n, t_2)) = 1 + \mathsf{nodes}(t_1) + \mathsf{nodes}(t_2)$.
- The function $\mathsf{sum} : \mathsf{BinTree} \to \mathsf{Nat}$ that maps a binary tree to the sum of the natural numbers attached to its nodes is defined by pattern matching as:
  1. $\mathsf{sum}(\mathsf{Leaf}(n)) = n$.
  2. $\mathsf{sum}(\mathsf{Branch}(t_1, n, t_2)) = n + \mathsf{sum}(t_1) + \mathsf{sum}(t_2)$.
- The function $\mathsf{ht} : \mathsf{BinTree} \to \mathsf{Nat}$ that maps a binary tree to its height is defined by pattern matching as:
  1. $\mathsf{ht}(\mathsf{Leaf}(n)) = 0$.
  2. $\mathsf{ht}(\mathsf{Branch}(t_1, n, t_2)) = 1 + \mathsf{max}(\mathsf{ht}(t_1), \mathsf{ht}(t_2))$.

Binary Trees (iClicker)

Let t be a member of BinTree. Which of the following formulas is true?

A. $\mathrm{nodes}(t) = 2^{\mathrm{ht}(t)}$.

B. $\mathrm{nodes}(t) \leq 2^{\mathrm{ht}(t)}$.

C. $\mathrm{nodes}(t) = 2^{\mathrm{ht}(t)+1} - 1$.

D. $\boxed{\mathrm{nodes}(t) \leq 2^{\mathrm{ht}(t)+1} - 1.}$

Example 2: Structural Induction for BinTree

Theorem. $\forall\, t \in \mathsf{BinTree}\, .\, \mathsf{nodes}(t) \leq 2^{\mathsf{ht}(t)+1} - 1.$

Proof. Let $P(t) \equiv \mathsf{nodes}(t) \leq 2^{\mathsf{ht}(t)+1} - 1$. We will prove $P(t)$ for all $t \in \mathsf{BinTree}$ by structural induction.

Base case: $t = \mathsf{Leaf}(n)$. We must show $P(t)$. $\mathsf{nodes}(t) = 1$ and $\mathsf{ht}(t) = 0$. So $\mathsf{nodes}(t) = 2^{\mathsf{ht}(t)+1} - 1$. Hence $P(t)$ holds.

Induction step: $t = \mathsf{Branch}(t_1, n, t_2)$. Assume $P(t_1)$ and $P(t_2)$. We must show $P(t)$.

$$
\begin{aligned}
\mathsf{nodes}(t) &= 1 + \mathsf{nodes}(t_1) + \mathsf{nodes}(t_2) && \langle \text{definition of nodes} \rangle \\
&\leq 1 + (2^{\mathsf{ht}(t_1)+1} - 1) + (2^{\mathsf{ht}(t_2)+1} - 1) && \langle \text{induction hypothesis} \rangle \\
&\leq 2 * 2^{\max(\mathsf{ht}(t_1),\mathsf{ht}(t_2))+1} - 1 && \langle \text{arithmetic} \rangle \\
&= 2^{1+\max(\mathsf{ht}(t_1),\mathsf{ht}(t_2))+1} - 1 && \langle \text{arithmetic} \rangle \\
&= 2^{\mathsf{ht}(t)+1} - 1 && \langle \text{definition of ht} \rangle
\end{aligned}
$$

Hence $P(t)$ holds.

Therefore, the theorem holds by structural induction.

Admin — January 15

- The following will be posted on Friday:
  1. 02 Exercises with Solutions.
  2. 03 Exercises.
  3. Assignment 1.
  4. Extra Credit Assignment 1.
- Friday will be the first discussion session.
- See the announcement on Avenue for the invitation to Discord and how to post M&Ms.
- All Questions Answers! is happening in next week's tutorials.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Review

- Inductive sets.
- Recursive definition with pattern matching.
- Structural induction.

3. Orders

Orders (iClicker)

Let (H, A) where H is a set of humans and A is a binary relation on H such that $h_1 \; A \; h_2$ means h_1 is an ancestor of h_2. (H, A) is

A. A weak partial order.

B. $\boxed{\text{A strict partial order.}}$

C. A weak total order.

D. A strict total order.

Pre-Orders

- A pre-order is a mathematical structure $(S, \leq)$ where $\leq$ is a binary relation on S that is:
  - Reflexive: $\forall x \in S \,.\, x \leq x$.
  - Transitive: $\forall x, y, z \in S \,.\, x \leq y \wedge y \leq z \Rightarrow x \leq z$.
- Example: $(F, \Rightarrow)$ is a pre-order where F is a set of formulas and $\Rightarrow$ is implication.
- A pre-order can have cycles.
- Every binary relation R on a set S can be extended to a pre-order on S by taking the reflexive and transitive closure of R.
- A equivalence relation is a pre-order (S, E) that is:
  - Symmetric: $\forall x, y \in S \,.\, x \, E \, y \Rightarrow y \, E \, x$.

Partial Orders

- A weak partial order is a mathematical structure $(S, \leq)$ where $\leq$ is a binary relation on S that is:
  - Reflexive: $\forall x \in S \,.\, x \leq x$.
  - Antisymmetric: $\forall x, y \in S \,.\, (x \leq y \wedge y \leq x) \Rightarrow x = y$.
  - Transitive: $\forall x, y, z \in S \,.\, (x \leq y \wedge y \leq z) \Rightarrow x \leq z$.
- A strict partial order is a mathematical structure $(S, <)$ where $<$ is a binary relation on S that is:
  - Irreflexive: $\forall x \in S \,.\, \neg(x < x)$.
  - Asymmetric: $\forall x, y \in S \,.\, x < y \Rightarrow \neg(y < x)$.
  - Transitive: $\forall x, y, z \in S \,.\, (x < y \wedge y < z) \Rightarrow x < z$.
- Examples: $(\mathcal{P}(S), \subseteq)$ and $(\mathcal{P}(S), \subset)$ are weak and strict partial orders.
- A partial order does not have cycles.
- Every pre-order can be interpreted as a partial order.

Weak vs. Strict Orders (iClicker)

If $(S, \leq)$ is a weak partial order, then $(S, <)$ will be the same order expressed as strict partial order if $<$ is defined as

A. $a < b$ iff $a \leq b \wedge a = b$.

B. $a < b$ iff $a \leq b \vee a = b$.

C. $\boxed{a < b \text{ iff } a \leq b \wedge a \neq b.}$

D. $a < b$ iff $a \leq b \vee a \neq b$.

Some Basic Order Definitions

- Let $(P, \leq)$ be a weak partial order and $S \subseteq P$.
- A maximal element [minimal element] of S is a $M \in S$ [$m \in S$] such that $\neg(M < x)$ [$\neg(x < m)$] for all $x \in S$.
- The maximum element or greatest element [minimum element or least element] of S, if it exists, is the $M \in S$ [$m \in S$] such that $x \leq M$ [$m \leq x$] for all $x \in S$.
- An upper bound [lower bound] of S is a $u \in P$ [$l \in P$] such that $x \leq u$ [$l \leq x$] for all $x \in S$.
- The least upper bound or supremum [greatest lower bound or infimum of S, if it exists, is a $U \in P$ [$L \in P$] such that U is an upper bound of S and, if u is an upper bound of S, then $U \leq u$ [L is a lower bound of S and, if l is a lower bound of S, then $l \leq L$].

Total Orders

- A weak total order is a mathematical structure $(S, \leq)$ where $\leq$ is a binary relation on S that is:
  - (Reflexive): $\forall x \, . \, x \leq x$.
  - Antisymmetric: $\forall x, y \in S \, . \, x \leq y \wedge y \leq x \Rightarrow x = y$.
  - Transitive: $\forall x, y, z \in S \, . \, x \leq y \wedge y \leq z \Rightarrow x \leq z$.
  - Total: $\forall x, y \in S \, . \, x \leq y \vee y \leq x$.
- A strict total order is a mathematical structure $(S, <)$ where $<$ is a binary relation on S that is:
  - Irreflexive: $\forall x \in S \, . \, \neg(x < x)$.
  - (Asymmetric): $\forall x, y \in S \, . \, x < y \Rightarrow \neg(y < x)$.
  - Transitive: $\forall x, y, z \in S \, . \, x < y \wedge y < z \Rightarrow x < z$.
  - Trichotomous: $\forall x, y \in S \, . \, x < y \vee y < x \vee x = y$.

  Examples: $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{Q}, \leq)$, and $(\mathbb{R}, \leq)$ are weak total orders.

Well-Orders

- A well-order is a strict total order $(S, <)$ such that every nonempty subset of S has a minimum element with respect to $<$.
- Proposition. Every well-order $(S, <)$ is Noetherian, i.e., S contains no infinite descending sequences of the form

$$\cdots < x_2 < x_1 < x_0.$$

- Examples.
  1. $(\mathbb{N}, <)$ is a well-order where $<$ is the usual order on $\mathbb{N}$.
  2. $(\mathbb{Z}, <)$ is **not** a well-order where $<$ is the usual order on $\mathbb{Z}$.
  3. $(S, <)$ is a well-order where S is

     $$\{0, \tfrac{1}{2}, \tfrac{2}{3}, \tfrac{3}{4}, \ldots, 1, 1\tfrac{1}{2}, 1\tfrac{2}{3}, 1\tfrac{3}{4}, \ldots\}$$

     and $<$ is the usual order on $\mathbb{Q}$.
  4. $(\mathbb{N} \times \mathbb{N}, <_{\mathsf{lex}})$ is a well-order where $<_{\mathsf{lex}}$ is lexicographic order on $\mathbb{N} \times \mathbb{N}$.

4. Ordinal Recursion and Induction

Ordinal Recursion

- Let $(S, <)$ be a well-order.
- A function $f : I \to O$ is defined by ordinal recursion on $(S, <)$ as follows:
  - The definition of f has the form
    $$f(x) = E(f(a_0(x)), \ldots, f(a_n(x))).$$
  - Each $i \in I$ is assigned a complexity $c(i) \in S$.
  - For all $i \in I$ and $m \in \mathbb{N}$ with $0 \leq m \leq n$,
    $$c(a_m(i)) < c(i).$$
- Example. Ordinal recursion on $(\mathbb{N}, <)$ is natural number recursion.
- Ordinal recursion is also called well-ordered recursion.

Ordinal Induction

- Let $(S, <)$ be a well-order.
- The ordinal induction principle for $(S, <)$ is:

  $$\begin{array}{l}\forall x \in S \,.\, ((\forall y \in S \,.\, y < x \Rightarrow P(y)) \Rightarrow P(x)) \\ \quad \Rightarrow \forall x \in S \,.\, P(x)\end{array}$$

  holds for every property P of S.
- Example. The ordinal induction principle for $(\mathbb{N}, <)$ is:

  $$\begin{array}{l}\forall x \in \mathbb{N} \,.\, ((\forall y \in \mathbb{N} \,.\, y < x \Rightarrow P(y)) \Rightarrow P(x)) \\ \quad \Rightarrow \forall x \in \mathbb{N} \,.\, P(x)\end{array}$$

  holds for every property P of $\mathbb{N}$.
  This principle is strong induction!
- Ordinal induction is also called well-ordered induction.
- Ordinal induction is useful since complicated nested induction arguments can be expressed using the ordinal induction principle for $(\gamma, <)$ for a suitable ordinal γ.

Ordinals (Optional) [1/3]

- An ordinal is a set γ such that $(\gamma, \in)$ is a well-order and, for all $\alpha \in \gamma$, $\alpha \subseteq \gamma$.
- Proposition.

  1. If β is an ordinal, then $\alpha \in \beta$ implies α is an ordinal.
  2. If α and β are ordinals, then $\alpha \subset \beta$ implies $\alpha \in \beta$.

- For ordinals α and β, let $\alpha < \beta$ mean $\alpha \in \beta$.
- The natural numbers are the finite ordinals where $0 = \emptyset$ and $n + 1 = n \cup \{n\}$.
- $\omega = \{0, 1, 2, \ldots\} = \mathbb{N}$ is the first infinite ordinal.
  - The ordinal induction principle for $(\omega, <)$ is strong induction.
  - The ordinal induction principle for $(\gamma, <)$ is an instance of transfinite induction when $\omega < \gamma$.

Ordinals (Optional) [2/3]

- $(\mathbb{O}, <)$ is a well-order where $\mathbb{O}$ is the collection of ordinals.
  - $\mathbb{N} \subseteq \mathbb{O} \subseteq \mathbb{S}$ where $\mathbb{S}$ is the surreal numbers.
  - $(\mathbb{O}, 0, +, *)$ is a near-semiring algebraic structure.
- Theorem. For every well-order $(S, <)$, there is some $\gamma \in \mathbb{O}$ such that $(S, <)$ and $(\gamma, <)$ are isomorphic.
  - γ is called the order type of $(S, <)$.

Ordinals (Optional) [3/3]

Examples of order types:

1. $\omega = \{0, 1, 2, 3, \ldots\}$ is the order type of $(\mathbb{N}, <)$.
2. $\omega + \omega = \{0, 1, 2, 3, \ldots, \omega + 0, \omega + 1, \omega + 2, \omega + 3, \ldots\}$ is the order type of $(S, <)$ where S is
$$\{0, \tfrac{1}{2}, \tfrac{2}{3}, \tfrac{3}{4}, \ldots, 1, 1\tfrac{1}{2}, 1\tfrac{2}{3}, 1\tfrac{3}{4}, \ldots\}.$$
3. $\omega * \omega = \cup\{\omega * 0, \omega * 1, \omega * 2, \omega * 3, \ldots\}$ is the order type of $(\mathbb{N} \times \mathbb{N}, <_{\mathrm{lex}})$.
4. $\omega^{\omega} = \cup\{\omega^0, \omega^1, \omega^2, \omega^3, \ldots\}$ is the order type of $(S, <)$ where S is the set of base ω numerals and $<$ is base ω numeric order.
5. $\epsilon_0 = \cup\{1, \omega, \omega^{\omega}, \omega^{\omega^{\omega}}, \ldots\}$ is the order type of $(S, <)$ where S is the set of finite rooted trees and $<$ is a kind of lexicographic order. Note: $\epsilon_0 = \omega^{\epsilon_0}$.

Example: Ackermann Function

- A leading version of the Ackermann function is

  $$A : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$

  recursively defined by:

  $$\begin{aligned} &A(0, n) = n + 1. \\ &A(m, 0) = A(m-1, 1) \text{ if } m > 0. \\ &A(m, n) = A(m-1, A(m, n-1)) \text{ if } m, n > 0. \end{aligned}$$

- Theorem. A is not primitive recursive.
  Proof. Show A grows faster than every primitive recursive function.
- Theorem. A is a total computable function.
  Proof. By ordinal induction on $(\mathbb{N} \times \mathbb{N}, <_{\mathrm{lex}})$.
- A is an extremely rapid growing function!
  - For example, $A(4, 3) = 2^{2^{65538}} - 3 = 2^{2^{2^{2^{2^{2}}}}} - 3$.

Admin — January 21

- The following will be posted on Friday:
  1. 03 Exercises with Solutions.
  2. 04 Exercises.
  3. Assignment 2.
- Assignment 1.

  Question 1. Prove

  $$\prod_{i=1}^{n} \frac{i^2}{i+1} = \frac{\mathrm{fact}(n)}{n+1}.$$

  for all $n \in \mathbb{N}$.

  Question 2. Prove $\text{base-2-exp}(n) = 2^n$ for all $n \in \mathbb{N}$.
- All Questions Answers! this week in the tutorials.
- Office hours: To see me please send me a note with times.
- Are there any questions?

Issues Mentioned in the Week 02 M&Ms

- Recursion and complexity.
- Recursion vs. induction.
- Inductive sets.
- Orders.
- Ordinals.

Review

- Orders
- Ordinal recursion and induction.

5. Well-Founded Recursion and Induction

Well-Founded Relations

- Let $R \subseteq U \times U$.
- y is an R-minimal element of $S \subseteq U$ if $y \in S$ and $\forall x \in U \,.\, x \in S \Rightarrow \neg(x \mathrel{R} y)$.
- R is well-founded if every nonempty subset of U has an R-minimal element.
- (U, R) is well-founded if R is well-founded.
- Examples:
  - $(\mathbb{N}, <)$.
  - $(\mathbb{N} \times \mathbb{N}, <_{\mathrm{lex}})$.
  - $(\mathbb{N}, R_{\mathsf{suc}})$ where $m \mathrel{R_{\mathsf{suc}}} n$ iff $n = m + 1$.

Noetherian Structures

- Let $R \subseteq U \times U$.
- A sequence $\langle x_0, x_1, x_2, \ldots \rangle$ of members of U is a descending R-sequence if

$$\cdots x_2 \; R \; x_1 \; R \; x_0.$$

- (U, R) is Noetherian if every descending R-sequence of members of U is finite.
- Proposition. (U, R) is well-founded iff (U, R) is Noetherian.

Well-Founded Relations (iClicker)

If $R \subseteq U \times U$ is a well-founded relation, then R need not be

A. Irreflexive.

B. Asymmetric.

C. $\boxed{\text{Transitive.}}$

D. None of the above.

Well-Founded Recursion and Induction [1/2]

- Let (U, R) be well-founded.
- A function $f : I \to O$ is defined by well-founded recursion on (U, R) as follows:
  - The definition of f has the form
    $$f(x) = E(f(a_0(x)), \ldots, f(a_n(x))).$$
  - Each $i \in I$ is assigned a complexity $c(i) \in U$.
  - For all $i \in I$ and $m \in \mathbb{N}$ with $0 \leq m \leq n$,
    $$c(a_m(i)) \mathrel{R} c(i).$$
- The well-founded induction principle for (U, R) is:
  $$\begin{array}{l}\forall\, x \in U \,.\, ((\forall\, y \in U \,.\, y \mathrel{R} x \Rightarrow P(y)) \Rightarrow P(x)) \\ \quad \Rightarrow \forall\, x \in U \,.\, P(x)\end{array}$$

  holds for every property P of U.

Well-Founded Recursion and Induction [2/2]

- Well-founded recursion and induction generalizes:
  1. Natural number recursion and induction.
  2. Structure recursion and induction.
  3. Ordinal recursion and induction.
- Proposition. If (U, R) is a strict total order, then (U, R) is well-founded iff (U, R) is a well-order.

Examples: Well-Founded Induction for $\mathbb{N}$

- The well-founded induction principle for $(\mathbb{N}, <)$ is:

  $$\forall x \in \mathbb{N} \,.\, ((\forall y \in \mathbb{N} \,.\, y < x \Rightarrow P(y)) \Rightarrow P(x)) \\ \Rightarrow \forall x \in \mathbb{N} \,.\, P(x)$$

  holds for every property P of $\mathbb{N}$.
  This is identical to strong induction!

- The well-founded induction principle for $(\mathbb{N}, R_{\mathrm{suc}})$ is:

  $$\forall x \in \mathbb{N} \,.\, ((\forall y \in \mathbb{N} \,.\, y \; R_{\mathrm{suc}} \; x \Rightarrow P(y)) \Rightarrow P(x)) \\ \Rightarrow \forall x \in \mathbb{N} \,.\, P(x)$$

  holds for every property P of $\mathbb{N}$.
  This is essentially the same as weak induction!

6. Summary

Which Induction Principle should Your Use

- Use natural number induction (weak or strong) to prove a statement that involves natural numbers or that requires a simple induction argument.
- Use structural induction to prove statements in which the underlying values are members of an inductive set.
- Use ordinal induction to prove a statement that involves a well-order or that requires a complex induction argument.
- Use well-founded induction to prove statements that involve a well-founded relation that is not a total order.

Strengthening the Induction Hypothesis

- Sometimes a statement cannot be proved by induction because the the resulting induction hypothesis is too weak.
- The strategy of strengthening the induction hypothesis is to prove a stronger statement that results in a stronger induction hypothesis.
- Example: Every square number is the sum of two triangle numbers.
  - This cannot be directly proved by induction because the induction hypothesis is too weak.
  - The stronger statement — "Every square number is the sum of two consecutive triangle numbers." — can be proved by induction because the induction hypothesis is stronger.