

COSE212: Programming Languages

Lecture 2 — Inductive Definitions (2)

Hakjoo Oh
2026 Fall

Contents

- More examples of inductive definitions
 - ▶ natural numbers, strings, booleans
 - ▶ lists, binary trees
 - ▶ arithmetic expressions, propositional logic
- Structural induction
 - ▶ three example proofs

Natural Numbers

The set of natural numbers:

$$\mathbb{N} = \{0, 1, 2, 3, \dots\}$$

is inductively defined:

$$\overline{0} \quad \frac{n}{n+1}$$

The inference rules can be expressed by a grammar:

$$n \rightarrow 0 \mid n + 1$$

Interpretation:

- 0 is a natural number.
- If n is a natural number then so is $n + 1$.

Strings

The set of strings over alphabet $\{a, \dots, z\}$, e.g., ϵ , a , b , $\dots$, z , aa , ab , $\dots$, az , ba , $\dots$, az , aaa , $\dots$, zzz , and so on. Inference rules:

$$\frac{}{\epsilon} \quad \frac{\alpha}{a\alpha} \quad \frac{\alpha}{b\alpha} \quad \dots \quad \frac{\alpha}{z\alpha}$$

or simply,

$$\frac{}{\epsilon} \quad \frac{\alpha}{x\alpha} \quad x \in \{a, \dots, z\}$$

In grammar:

$$\begin{array}{c} \alpha \rightarrow \epsilon \\ | \quad x\alpha \quad (x \in \{a, \dots, z\}) \end{array}$$

Boolean Values

The set of boolean values:

$$\mathbb{B} = \{\textit{true}, \textit{false}\}.$$

If a set is finite, just enumerate all of its elements by axioms:

$$\overline{\textit{true}} \quad \overline{\textit{false}}$$

In grammar:

$$b \rightarrow \textit{true} \mid \textit{false}$$

Lists

Examples of lists of integers:

- ① **nil**
- ② **14 · nil**
- ③ **3 · 14 · nil**
- ④ **−7 · 3 · 14 · nil**

Inference rules:

$$\frac{}{\text{nil}} \quad \frac{l}{n \cdot l} \quad n \in \mathbb{Z}$$

In grammar:

$$l \rightarrow \begin{array}{l} \text{nil} \\ n \cdot l \end{array} \quad (n \in \mathbb{Z})$$

Lists

A proof that $-7 \cdot 3 \cdot 14 \cdot \mathbf{nil}$ is a list of integers:

$$\frac{\frac{\frac{\overline{\mathbf{nil}}}{14 \cdot \mathbf{nil}} \quad 14 \in \mathbb{Z}}{3 \cdot 14 \cdot \mathbf{nil}} \quad 3 \in \mathbb{Z}}{-7 \cdot 3 \cdot 14 \cdot \mathbf{nil}} \quad -7 \in \mathbb{Z}$$

The proof tree is also called *derivation tree* or *deduction tree*.

Binary Trees

Examples of binary trees:

- ① **leaf**
- ② **(2, leaf, leaf)**
- ③ **(1, (2, leaf, leaf), leaf)**
- ④ **(1, (2, leaf, leaf), (3, (4, leaf, leaf), leaf))**

Inference rules:

$$\overline{\text{leaf}} \quad \frac{t_1 \quad t_2}{(n, t_1, t_2)} \quad n \in \mathbb{Z}$$

In grammar:

$$\begin{array}{c} t \rightarrow \text{leaf} \\ | \quad (n, t, t) \quad (n \in \mathbb{Z}) \end{array}$$

Binary Trees

A proof that

$(1, (2, \text{leaf}, \text{leaf}), (3, (4, \text{leaf}, \text{leaf}), \text{leaf}))$

is a binary tree:

$$\frac{\frac{\overline{\text{leaf}}}{(2, \text{leaf}, \text{leaf})} \quad 2 \in \mathbb{Z} \quad \frac{\frac{\overline{\text{leaf}}}{(4, \text{leaf}, \text{leaf})} \quad 4 \in \mathbb{Z}}{(3, (4, \text{leaf}, \text{leaf}), \text{leaf})} \quad 3 \in \mathbb{Z}}{(1, (2, \text{leaf}, \text{leaf}), (3, (4, \text{leaf}, \text{leaf}), \text{leaf}))} \quad 1 \in \mathbb{Z}$$

Binary Trees: a different version

Binary tree examples: 1, (1, nil), (1, 2), ((1, 2), nil), ((1, 2), (3, 4)).

Inference rules:

$$\overline{n} \quad n \in \mathbb{Z} \qquad \frac{t}{(t, \text{nil})} \qquad \frac{t}{(\text{nil}, t)} \qquad \frac{t_1 \quad t_2}{(t_1, t_2)}$$

In grammar:

$$\begin{array}{lcl} t & \rightarrow & n \quad (n \in \mathbb{Z}) \\ & & | \quad (t, \text{nil}) \\ & & | \quad (\text{nil}, t) \\ & & | \quad (t, t) \end{array}$$

A proof that ((1, 2), (3, nil)) is a binary tree:

$$\frac{\frac{\overline{1} \quad \overline{2}}{(1, 2)} \quad \frac{\overline{3}}{(3, \text{nil})}}{((1, 2), (3, \text{nil}))}$$

Expressions

Expression examples: 2 , -2 , $1 + 2$, $1 + (2 * (-3))$, etc.

Inference rules:

$$\overline{n} \quad n \in \mathbb{Z} \qquad \frac{e}{-e} \qquad \frac{e_1 \quad e_2}{e_1 + e_2} \qquad \frac{e_1 \quad e_2}{e_1 * e_2} \qquad \frac{e}{(e)}$$

In grammar:

$$\begin{array}{lcl} e & \rightarrow & n \quad (n \in \mathbb{Z}) \\ & | & -e \\ & | & e + e \\ & | & e * e \\ & | & (e) \end{array}$$

Example:

$$\frac{\overline{1} \quad \frac{\overline{2} \quad \frac{\overline{3} \quad \overline{-3}}{\overline{-3}}}{\overline{2 * (-3)}}}{\overline{1 * (-3)}}}{\overline{1 + (2 * (-3))}}$$

Propositional Logic

Examples:

- T, F
- $T \wedge F$
- $T \vee F$
- $(T \wedge F) \wedge (T \vee F)$
- $T \Rightarrow (F \Rightarrow T)$

Propositional Logic

Syntax:

$$\begin{array}{lcl} f & \rightarrow & T \mid F \\ & | & \neg f \\ & | & f \wedge f \\ & | & f \vee f \\ & | & f \Rightarrow f \end{array}$$

Semantics ($\llbracket f \rrbracket$):

$$\begin{array}{ll} \llbracket T \rrbracket & = \text{true} \\ \llbracket F \rrbracket & = \text{false} \\ \llbracket \neg f \rrbracket & = \text{not } \llbracket f \rrbracket \\ \llbracket f_1 \wedge f_2 \rrbracket & = \llbracket f_1 \rrbracket \text{ andalso } \llbracket f_2 \rrbracket \\ \llbracket f_1 \vee f_2 \rrbracket & = \llbracket f_1 \rrbracket \text{ orelse } \llbracket f_2 \rrbracket \\ \llbracket f_1 \Rightarrow f_2 \rrbracket & = \llbracket f_1 \rrbracket \text{ implies } \llbracket f_2 \rrbracket \end{array}$$

Propositional Logic

$$\begin{aligned}\llbracket (T \wedge (T \vee F)) \Rightarrow F \rrbracket &= \llbracket T \wedge (T \vee F) \rrbracket \text{ implies } \llbracket F \rrbracket \\ &= (\llbracket T \rrbracket \text{ andalso } \llbracket T \vee F \rrbracket) \text{ implies } \textit{false} \\ &= (\textit{true} \text{ andalso } (\llbracket T \rrbracket \text{ orelse } \llbracket F \rrbracket)) \text{ implies } \textit{false} \\ &= (\textit{true} \text{ andalso } (\textit{true} \text{ orelse } \textit{false})) \text{ implies } \textit{false} \\ &= \textit{false}\end{aligned}$$

Structural Induction

A technique for proving properties about inductively defined sets.

To prove that a proposition $P(s)$ is true for all structures s , prove the following:

- 1 (Base case) P is true on simple structures (those without substructures)
- 2 (Inductive case) If P is true on the substructures of s , then it is true on s itself. The assumption is called *induction hypothesis (I.H.)*.

Example 1

Let S be the set defined by the following inference rules:

$$\frac{}{3} \quad \frac{x \quad y}{x + y}$$

Prove that for all $x \in S$, x is divisible by 3.

Proof. By structural induction.

- (Base case) The base case is when x is 3. Obviously, x is divisible by 3.
- (Inductive case) The induction hypothesis (I.H.) is

$$x \text{ is divisible by } 3, \quad y \text{ is divisible by } 3.$$

Let $x = 3k_1$ and $y = 3k_2$. Using I.H., we derive

$$x + y \text{ is divisible by } 3$$

as follows:

$$\begin{aligned} x + y &= 3k_1 + 3k_2 \quad \dots \text{by I.H.} \\ &= 3(k_1 + k_2) \end{aligned}$$



Example 2

Let S be the set defined by the following inference rules:

$$\frac{}{()} \quad \frac{x}{(x)} \quad \frac{x \quad y}{xy}$$

Prove that every element of the set has the same number of '('s and ')'s.

Proof Restate the claim formally:

$$\text{If } x \in S \text{ then } l(x) = r(x)$$

where $l(x)$ and $r(x)$ denote the number of '('s and ')'s, respectively:

$$\begin{array}{ll} l(()) = 1 & r(()) = 1 \\ l((x)) = l(x) + 1 & r((x)) = r(x) + 1 \\ l(xy) = l(x) + l(y) & r(xy) = r(x) + r(y) \end{array}$$

We prove it by structural induction:

- (Base case): The base case is when $x = ()$. Then $l(x) = 1 = r(x)$.

Example 2

- (Inductive case): There are two inductive cases:

$$\frac{x}{(x)} \qquad \frac{x \quad y}{xy}$$

Induction hypotheses (I.H.):

$$l(x) = r(x), \qquad l(y) = r(y).$$

- ▶ The first case. We prove $l((x)) = r((x))$:

$$\begin{aligned} l((x)) &= l(x) + 1 && \dots \text{by definition of } l((x)) \\ &= r(x) + 1 && \dots \text{by I.H.} \\ &= r((x)) && \dots \text{by definition of } r((x)) \end{aligned}$$

- ▶ The second case. We prove $l(xy) = r(xy)$:

$$\begin{aligned} l(xy) &= l(x) + l(y) && \dots \text{by definition of } l(xy) \\ &= r(x) + r(y) && \dots \text{by I.H.} \\ &= r(xy) && \dots \text{by definition of } r(xy) \end{aligned}$$



Example 3

Let T be the set of binary trees:

$$\frac{}{\text{leaf}} \quad \frac{t_1 \quad t_2}{(n, t_1, t_2)} \quad n \in \mathbb{Z}$$

Prove that for all such trees, the number of leaves is always one more than the number of internal nodes.

Proof. Restate the claim more formally:

$$\text{If } t \in T \text{ then } l(t) = i(t) + 1$$

where $l(t)$ and $i(t)$ denote the number of leaves and internal nodes, respectively:

$$\begin{array}{ll} l(\text{leaf}) &= 1 & i(\text{leaf}) &= 0 \\ l(n, t_1, t_2) &= l(t_1) + l(t_2) & i(n, t_1, t_2) &= i(t_1) + i(t_2) + 1 \end{array}$$

We prove it by structural induction:

- (Base case): The base case is when $t = \text{leaf}$, where $l(t) = 1$ and $i(t) = 0$.
- (Inductive case): The induction hypothesis:

$$l(t_1) = i(t_1) + 1, \quad l(t_2) = i(t_2) + 1$$

Using I.H., we prove $l((n, t_1, t_2)) = i((n, t_1, t_2)) + 1$:

$$\begin{aligned} l((n, t_1, t_2)) &= l(t_1) + l(t_2) && \text{definition of } l \\ &= i(t_1) + 1 + i(t_2) + 1 && \text{by induction hypothesis} \\ &= i(n, t_1, t_2) + 1 && \text{definition of } i \end{aligned}$$

Summary

- Computer science is full of inductive definitions.
 - ▶ primitive values: booleans, characters, integers, strings, etc
 - ▶ compound values: lists, trees, graphs, etc
 - ▶ language syntax and semantics
- Structural induction
 - ▶ a general technique for reasoning about inductively defined sets