Instructors: Kazutaka Matsuda and Eijiro Sumii

Inductive Definition

Inductive definition: a style of definition of a set S that consists of (1) rules saying "for any $s_1, \ldots, s_n$, $f(s_1, \ldots, s_n)$ belongs to S assuming $s_1, \ldots, s_n$ belong to S", and (2) "no other things belong to S". Sometimes, (2') "S is the smallest set satisfying (1)" is used instead of (2).

Commonly, we omit (2) or (2') by saying that "S is inductively defined as ...".

Example(s). The set of even numbers $\mathbb{N}_{\text{even}}$ is defined as follows.

- $0 \in \mathbb{N}_{\text{even}}$.
- $n + 2 \in \mathbb{N}_{\text{even}}$ for all $n \in \mathbb{N}_{\text{even}}$.
- No other numbers belong to $\mathbb{N}_{\text{even}}$.

Notice that, only by the first and second rule, the set $\mathbb{N}_{\text{even}}$ can contain 1; check that $\mathbb{N}_{\text{even}} = \mathbb{N}$ satisfies the first and second rule. □

Example(s). The set of even numbers $\mathbb{N}_{\text{even}}$ is defined *inductively* as follows.

- $0 \in \mathbb{N}_{\text{even}}$.
- $n + 2 \in \mathbb{N}_{\text{even}}$ for all $n \in \mathbb{N}_{\text{even}}$. □

Example(s). The set of binary trees $\mathcal{B}$ is defined inductively as follows.

- $\mathsf{leaf} \in \mathcal{B}$
- $\mathsf{node}(t_1, t_2) \in \mathcal{B}$ for all $t_1, t_2 \in \mathcal{B}$.

For an inductively defined set, we have the corresponding induction principle. For example, we have the following induction principles

Theorem (Induction Principle on Even Numbers)**.** For any unary predicate P,

$$(\forall n \in \mathbb{N}_{\text{even}}.\, P(n)) \quad \Leftrightarrow \quad P(0) \wedge (\forall n \in \mathbb{N}_{\text{even}}.\, P(n) \Rightarrow P(n+2))$$

holds. □

Theorem (Induction Principle on Binary Trees)**.** For any unary predicate P,

$$(\forall t \in \mathcal{B}.\, P(t)) \quad \Leftrightarrow \quad P(\mathsf{leaf}) \wedge (\forall t_1, t_2 \in \mathcal{B}.\, P(t_1) \wedge P(t_2) \Rightarrow P(\mathsf{node}(t_1, t_2)))$$

holds. □

Exercise. Let $\mathit{leaves}(t)$ be the number of leaves in t and $\mathit{nodes}(t)$ be the number of nodes in t. Prove by induction that $\mathit{leaves}(t) = \mathit{nodes}(t) + 1$ for any $t \in \mathcal{B}$. □

Inductive Definition of Functions and Relations

We can define functions and relations inductively. They are just special cases of sets.

Example(s). We define the subtree relation $\preceq$ on binary trees inductively as follows.

- $\mathsf{leaf} \preceq \mathsf{leaf}$.
- $t \preceq \mathsf{node}(t_1, t_2)$ if either $t = \mathsf{node}(t_1, t_2)$ or $t \preceq t_1$ or $t \preceq t_2$ for any $t, t_1, t_2 \in \mathcal{B}$. □

Example(s). We define the function *leaves* that computes the number of leaves, inductively as follows.

- $\mathit{leaves}(\mathsf{leaf}) = 1$.
- $\mathit{leaves}(\mathsf{node}(t_1, t_2)) = \mathit{leaves}(t_1) + \mathit{leaves}(t_2)$, for any $t_1, t_2 \in \mathcal{B}$. □

Exercise. Define the function *nodes* inductively. How about the function *height* that computes the length of the longest path from the root to a leaf, where $\mathit{height}(\mathsf{leaf}) = 0$. □

Exercise. Prove by induction that $t_1 \preceq t_2$ implies $\mathit{nodes}(t_1) \leq \mathit{nodes}(t_2)$ for all $t_1, t_2 \in \mathcal{B}$.

FYI: Mathematical Foundation on Inductive Definition

Definition. For a set S, a function $f : 2^S \to 2^S$ is *monotone* if $X \subseteq Y$ implies $f(X) \subseteq f(Y)$ for all $X, Y \in 2^S$. □

Definition. For a function f, $x \in \mathrm{dom}(f)$ is called a *fixed point* if $x = f(x)$. □

Example(s). Let f_{even} be a function defined by $f_{\mathrm{even}}(X) = \{0\} \cup \{n+2 \mid n \in X\}$. Then, the set of even numbers $\mathbb{N}_{\mathrm{even}}$ and the set of natural numbers $\mathbb{N}$ are only fixed points of f_{even}. □

Theorem ((An Instance of) Tarski's Fixpoint Theorem)**.** For any monotone function $f : 2^S \to 2^S$, the least fixed point of f exists and is given by $\bigcap\{Y \mid f(Y) \subseteq Y\}$. □

Example(s). For f_{even}, $f_{\mathrm{even}}(Y) \subseteq Y$ says that $0 \in Y$ and $n+2 \in Y$ for any $n \in Y$. The set $\bigcap\{Y \mid f_{\mathrm{even}}(Y) \subseteq Y\}$ is the smallest one that satisfies this condition, and thus nothing but the inductive definition of $\mathbb{N}_{\mathrm{even}}$ itself. □

Corollary. Let $f : 2^S \to 2^S$ be a monotone function, and X be its least fixed point. For any set Y satisfying $f(Y) \subseteq Y$, $X \subseteq Y$ holds. □

Note. Let $Y = \{x \in \mathbb{N}_{\mathrm{even}} \mid P(x)\}$ for a unary predicate P. Then, showing $f_{\mathrm{even}}(Y) \subseteq Y$ is nothing but showing $P(0) \wedge (\forall n \in \mathbb{N}_{\mathrm{even}}.\, P(n) \Rightarrow P(n+2))$. Thus, the corollary gives nothing but the induction principle on $\mathbb{N}_{\mathrm{even}}$.

Inference Rules

Inference rule: a rule written of the form of

$$\frac{A_1 \quad A_2 \quad \dots \quad A_n}{B}$$

that means "if A_1, A_2, ..., A_n hold, then B does". Sometimes the bar is omitted if there are no premises $A_1, A_2, \dots, A_n$.

Example(s). The set of even numbers $\mathbb{N}_{\text{even}}$ is defined by the following inference rules.

$$\frac{}{0 \in \mathbb{N}_{\text{even}}} \qquad \frac{n \in \mathbb{N}_{\text{even}}}{n + 2 \in \mathbb{N}_{\text{even}}}$$

The second rule contains the (meta-)variable[1] n that will be replaced by concrete numbers. Precisely speaking, this kind of rules are inference rule schemas rather than rules.

Similarly, we can define the set of binary trees $\mathcal{B}$ using inference rules as follows.

$$\frac{}{\mathsf{leaf} \in \mathcal{B}} \qquad \frac{t_1 \in \mathcal{B} \qquad t_2 \in \mathcal{B}}{\mathsf{node}(t_1, t_2) \in \mathcal{B}}$$

We need not name the set of binary trees to define the set of binary trees.

$$\frac{}{\mathsf{leaf}\ \textbf{binary-tree}} \qquad \frac{t_1\ \textbf{binary-tree} \qquad t_2\ \textbf{binary-tree}}{\mathsf{node}(t_1, t_2)\ \textbf{binary-tree}}$$

Here, t **binary-tree** is a judgment that states "t is a binary tree". □

Derivation tree: a tree of which every node is an instance of some inference rule. The existence of a derivation tree means that the premises of all the inference rules occurring in the tree are fulfilled, and thus we obtain the conclusion of its root.

Example(s). We conclude $4 \in \mathbb{N}_{\text{even}}$ and $\mathsf{node}(\mathsf{node}(\mathsf{leaf}, \mathsf{leaf}), \mathsf{leaf}) \in \mathcal{B}$ because we have the following derivation trees.

$$\frac{\dfrac{\dfrac{}{0 \in \mathbb{N}_{\text{even}}}}{2 \in \mathbb{N}_{\text{even}}}}{4 \in \mathbb{N}_{\text{even}}} \qquad \frac{\dfrac{\dfrac{}{\mathsf{leaf} \in \mathcal{B}} \qquad \dfrac{}{\mathsf{leaf} \in \mathcal{B}}}{\mathsf{node}(\mathsf{leaf}, \mathsf{leaf}) \in \mathcal{B}} \qquad \dfrac{}{\mathsf{leaf} \in \mathcal{B}}}{\mathsf{node}(\mathsf{node}(\mathsf{leaf}, \mathsf{leaf}), \mathsf{leaf}) \in \mathcal{B}}$$

Instead, we cannot conclude $3 \in \mathbb{N}_{\text{even}}$ or $\mathsf{node} \in \mathcal{B}$ because we do not have any derivation tree whose root concludes these statements. □

[1] Metavariables are just variables in mathematics. We usually use the term "variables" for variables in a target programming language.

Backus Naur Form (BNF)

BNF: A way to specify the syntax of a language as a context-free grammar. The following is an example.

$$\begin{array}{rcl} \langle\text{binary tree}\rangle & ::= & \mathsf{leaf} \\ & | & \mathsf{node}(\langle\text{binary tree}\rangle, \langle\text{binary tree}\rangle) \end{array}$$

One familiar with context-free grammars would find that this definition is similar to the following production rules.

$$\begin{array}{rcl} \langle\text{binary tree}\rangle & \to & \mathsf{leaf} \\ \langle\text{binary tree}\rangle & \to & \mathsf{node}(\langle\text{binary tree}\rangle, \langle\text{binary tree}\rangle) \end{array}$$

However, nowadays in the context of the programming language, maybe since our interests would not be mainly on string representations but on (abstract) syntax trees, the original-style BNF is less commonly used. Instead, we just use BNFs to define tree-like things inductively. Also, we do not use the special forms for nonterminals. Instead, we usually write either of the following style.

$$\begin{array}{rcl} t & ::= & \mathsf{leaf} \\ & | & \mathsf{node}(t_1, t_2) \end{array} \quad \text{or} \quad \begin{array}{rcl} t & ::= & \mathsf{leaf} \\ & | & \mathsf{node}(t, t) \end{array}$$

All the three different styles of the definition of binary trees define the same thing.

Example(s) (Propositional Formulas)**.** We define the set of propositional formulas by using the following BNF.

$$A, B ::= P \mid \neg A \mid A \wedge B \mid A \vee B \mid A \Rightarrow B$$

Here, P represents a propositional variable. □

Now, we are ready to define the syntax of the (untyped) λ-calculus!

$$M, N ::= x \mid \lambda x.M \mid M\ N$$

Here, x represents a variable. M and N are called λ-*terms* or λ-*expressions*.

Example(s). $\lambda x.x$, $\lambda x.y$, $\lambda x.(\lambda y.x)$, and $(\lambda x.(x\ x))\ (\lambda x.(x\ x))$ are examples of λ-terms. □

Structural Induction

Inductions for tree-like data such as those can be defined by BNFs sometimes are called *structural induction*.