4471028: 프로그래밍언어

Lecture 2 — 귀납 정의 (2)
Inductive Definition (2)

임현승
2020 봄학기

개요

- 판단, 추론 규칙, 문법
- 귀납 정의 예
  - 문자열
  - 불리언(Boolean)
  - 리스트
  - 이진 나무
  - 정수식
  - 명제 논리

자연수 집합을 정의해보자

자연수 집합:

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

추론 규칙을 이용하여 귀납적으로 정의하면

$$\frac{}{\mathtt{O} \in \mathbb{N}}\ \textsc{Zero} \qquad \frac{n \in \mathbb{N}}{\mathtt{S}\ n \in \mathbb{N}}\ \textsc{Succ}$$

다음과 같이 해석

$$\begin{array}{rcll} \mathtt{O} & \equiv & 0 & \\ \mathtt{S}\ \mathtt{O} & \equiv & 1 & \equiv 0+1 \\ \mathtt{S}\ \mathtt{S}\ \mathtt{O} & \equiv & 2 & \equiv 0+1+1 \\ \mathtt{S}\ \mathtt{S}\ \mathtt{S}\ \mathtt{O} & \equiv & 3 & \equiv 0+1+1+1 \\ & & \vdots & \\ \mathtt{S}\ n & \equiv & n+1 & \equiv 0\underbrace{+1+1+\cdots+1}_{n+1\text{ 개}} \end{array}$$

판단(Judgment)

- 판단 또는 판단문 $n \in \mathbb{N}$은 다음과 같이 해석

$$n \in \mathbb{N} \quad \Longleftrightarrow \quad \begin{array}{l} n \text{ is a natural number} \\ n \text{ belongs to the set } \mathbb{N} \\ n \text{ is in the set } \mathbb{N} \end{array}$$

- 판단은 증명할 수 있거나 증명 불가능한 지식의 대상 또는 객체 (an object of knowledge that may or may not be provable)

- 판단 예:
  - "$1 - 1$ is equal to 0."
  - "$1 + 1$ is equal to 0."
  - "It is raining."
  - "$0 + 1 + 1$ belongs to a set $\mathbb{N}$."

판단(Judgment)

- **계산(산술) 규칙**이 없다면 “$\mathbf{1} - \mathbf{1}$ is equal to $\mathbf{0}$”는 어떤 의미일까?
- 판단은 옳고 그름을 결정할 수 있는 추론 규칙(inference rules)이 있을 때에만 타당함
- 판단 $\mathbf{s}\ \mathbf{s}\ \mathbf{o} \in \mathbb{N}$ 을 증명하려면 추론 규칙을 이용하여 증명 나무(proof tree)를 완성해야 함

$$\cfrac{\cfrac{\cfrac{}{\mathbf{o} \in \mathbb{N}}\ \textsc{Zero}}{\mathbf{s}\ \mathbf{o} \in \mathbb{N}}\ \textsc{Succ}}{\mathbf{s}\ \mathbf{s}\ \mathbf{o} \in \mathbb{N}}\ \textsc{Succ}$$

- 증명 나무는 유도 나무(derivation tree) 또는 추론 나무(deduction tree)라고도 불림

문법(Grammar)

추론 규칙은 문법을 이용해서도 표현 가능:

$$\begin{array}{ll} \textsf{nat} & n ::= \mathsf{O} \\ & n ::= \mathsf{S}\ n \end{array}$$

- nat 문법을 이용하여 정의하고자 하는 구문의 범주(syntactic category) 또는 집합(set)의 이름, $\mathbb{N}$의 다른 이름
- n 비단말 기호(nonterminal symbol), 메타 변수(metavariable)
- O, S 단말 기호(terminal symbol), 프로그래밍 언어 토큰
- 생성 규칙(productions rules)
  (1) $n ::= \mathsf{O}$ (base case)
  (2) $n ::= \mathsf{S}\ n$ (inductive case)
  read $::=$ as "is (defined as)" or "can be"
  $::=$ 기호 대신에 $\rightarrow$ 기호를 이용하기도 함

문법(Grammar)

추론 규칙은 문법을 이용해서도 표현 가능:

$$\begin{array}{ll} \text{nat} & n ::= \mathsf{O} \\ & n ::= \mathsf{S}\ n \end{array}$$

해석:

- O is a natural number.
- If n is a natural number then so is $\mathsf{S}\ n$.

또는:

- A natural number n is either O or $\mathsf{S}\ n'$ for some another natural number n'.

여러 생성 규칙을 한 번에 작성하려면 | 를 이용

$$n ::= \mathsf{O} \mid \mathsf{S}\ n$$

read | as “or”.

문법으로부터 유도(Syntactic Derivation)

- 어떤 값 α가 집합 T의 원소임을 보이기 위해 증명 나무를 작성하는 대신에 집합 T의 원소를 정의하는 문법으로부터 α를 유도(derivation) 해도 된다.
- 유도는 주어진 문장(sentence, a string of nonterminal and terminal symbols)에서 사용되는 nonterminal symbol을 생성 규칙의 오른편에 있는 정의(definition)로 바꾸는 과정
- $\alpha \Rightarrow \beta$ 는 "문장 α는 문장 β를 유도한다"라고 읽음
- 생성 규칙 n ::= **s** n 을 이용하여 다음과 같은 유도가 가능

$$\mathbf{s}\ \mathbf{s}\ n \Rightarrow \mathbf{s}\ \mathbf{s}\ \mathbf{s}\ n$$

문법으로부터 유도(Syntactic Derivation)

$\mathtt{S\ S\ S\ O}$ 가 자연수임을 보이는 증명:

$$
\begin{array}{rcll}
n & \Rightarrow & \mathtt{S}\ n & \\
 & \Rightarrow & \mathtt{S\ S}\ n & \\
 & \Rightarrow & \mathtt{S\ S\ S}\ n & \\
 & \Rightarrow & \mathtt{S\ S\ S\ O} & \equiv \quad 3
\end{array}
$$

생성 규칙을 0번 이상 적용한 결과를 표기하려면 $\Rightarrow^*$(reflexive and transitive closure of $\Rightarrow$)를 이용

$$n \Rightarrow^* \mathtt{S\ S\ S\ O}$$

귀납 정의 예 - 문자열

알파벳(alphabet) $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$에 대한 문자열(string)들의 집합 T를 정의하라.

예: $\epsilon, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{aa}, \mathtt{ab}, \mathtt{ac}, \mathtt{ba}, \ldots, \mathtt{cc}, \mathtt{aaa}, \ldots, \mathtt{ccc}, \ldots$

판단문:

$$\alpha \ \ \mathsf{str} \quad \iff \quad \alpha \text{ is a string in } T$$

추론 규칙:

$$\frac{}{\epsilon \ \ \mathsf{str}} \qquad \frac{\alpha \ \ \mathsf{str}}{\mathtt{a}\alpha \ \ \mathsf{str}} \qquad \frac{\alpha \ \ \mathsf{str}}{\mathtt{b}\alpha \ \ \mathsf{str}} \qquad \frac{\alpha \ \ \mathsf{str}}{\mathtt{c}\alpha \ \ \mathsf{str}}$$

또는

$$\frac{}{\epsilon \ \ \mathsf{str}} \qquad \frac{\alpha \ \ \mathsf{str}}{x\alpha \ \ \mathsf{str}} \ (x \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\})$$

문법:

$$\begin{array}{lcll} \mathsf{str} \quad \alpha & ::= & \epsilon & \\ & \mid & x\alpha & (x \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}) \end{array}$$

불리언 값(Boolean Values)

불리언 값들의 집합:

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

집합의 원소의 개수가 유한하면, 집합의 원소를 모두 나열하면 됨.

판단문:

$$b \ \ \text{bool} \qquad \Longleftrightarrow \qquad b \text{ is a boolean value in } \mathbb{B}$$

추론 규칙:

$$\frac{}{true \ \ \text{bool}} \qquad \frac{}{false \ \ \text{bool}}$$

문법:

$$\text{bool} \qquad b ::= true \mid false$$

리스트(Lists)

정수들로 구성된 리스트의 예(OCaml 문법으로):

1. []
2. $\mathbf{14} :: []$
3. $\mathbf{3} :: \mathbf{14} :: []$
4. $-\mathbf{7} :: \mathbf{3} :: \mathbf{14} :: []$

판단문:

$$l \ \ \text{list} \quad \Longleftrightarrow \quad l \text{ is a list}$$

추론 규칙

$$\frac{}{[] \ \ \text{list}} \qquad \frac{l \ \ \text{list}}{n :: l \ \ \text{list}} \ (n \in \mathbb{Z})$$

문법:

$$\begin{array}{lcll} \text{list} \quad l & ::= & [] & \\ & \mid & n :: l & (n \in \mathbb{Z}) \end{array}$$

리스트(Lists)

$-7 :: 3 :: 14 :: []$ 가 정수로 구성된 리스트임을 증명:

$$
\cfrac{\cfrac{\cfrac{\overline{[]\ \ \mathsf{list}}}{14 :: []\ \ \mathsf{list}}\ (14 \in \mathbb{Z})}{3 :: 14 :: []\ \ \mathsf{list}}\ (3 \in \mathbb{Z})}{-7 :: 3 :: 14 :: []\ \ \mathsf{list}}\ (-7 \in \mathbb{Z})
$$

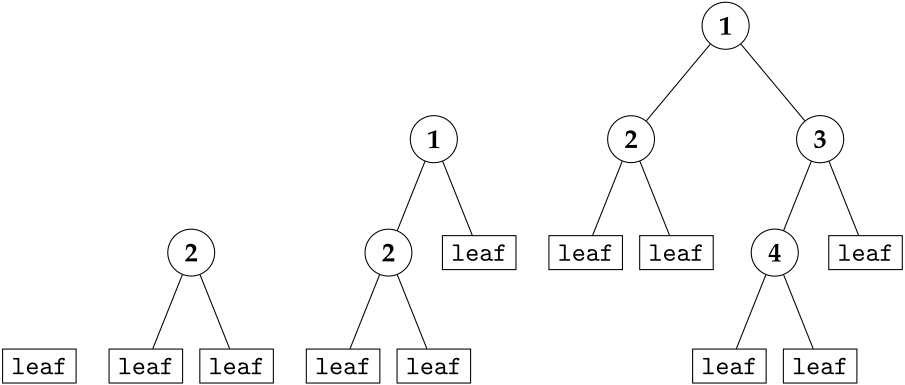
문법으로부터 유도해보면(유도 순서는 중요하지 않음):

$$
\begin{aligned}
l &\Rightarrow n :: l \\
&\Rightarrow n :: n :: l \\
&\Rightarrow n :: n :: n :: l \\
&\Rightarrow n :: n :: n :: [] \\
&\Rightarrow n :: n :: 14 :: [] \\
&\Rightarrow n :: 3 :: 14 :: [] \\
&\Rightarrow -7 :: 3 :: 14 :: []
\end{aligned}
$$

이진 나무(Binary Trees)

이진 나무 예:

- leaf
- $\texttt{node}(\mathbf{2}, \texttt{leaf}, \texttt{leaf})$
- $\texttt{node}(\mathbf{1}, \texttt{node}(\mathbf{2}, \texttt{leaf}, \texttt{leaf}), \texttt{leaf})$
- $\texttt{node}(\mathbf{1}, \texttt{node}(\mathbf{2}, \texttt{leaf}, \texttt{leaf}), \texttt{node}(\mathbf{3}, \texttt{node}(\mathbf{4}, \texttt{leaf}, \texttt{leaf}), \texttt{leaf}))$



이진 나무(Binary Trees)

판단문:

$$t \ \ \text{tree} \quad \iff \quad t \text{ is a binary tree}$$

추론 규칙:

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

문법:

$$\begin{array}{llll} \text{tree} & t & ::= & \texttt{leaf} \\ & & \mid & \texttt{node}(n, t, t) \quad (n \in \mathbb{Z}) \end{array}$$

이진 나무(Binary Trees)

다음 나무 구조가 이진 나무임을 증명하라.

$$\texttt{node}(\mathbf{1}, \texttt{node}(\mathbf{2}, \texttt{leaf}, \texttt{leaf}), \texttt{node}(\mathbf{3}, \texttt{node}(\mathbf{4}, \texttt{leaf}, \texttt{leaf}), \texttt{leaf}))$$

증명 나무:

$$\dfrac{\dfrac{\dfrac{}{\texttt{leaf}\ \ \mathsf{tree}} \quad \dfrac{}{\texttt{leaf}\ \ \mathsf{tree}}}{\texttt{node}(\mathbf{2}, \texttt{leaf}, \texttt{leaf})\ \ \mathsf{tree}}\ (\mathbf{2} \in \mathbb{Z}) \qquad \dfrac{\dfrac{\dfrac{}{\texttt{leaf}\ \ \mathsf{tree}} \quad \dfrac{}{\texttt{leaf}\ \ \mathsf{tree}}}{\texttt{node}(\mathbf{4}, \texttt{leaf}, \texttt{leaf})\ \ \mathsf{tree}}\ (\mathbf{4} \in \mathbb{Z}) \quad \dfrac{}{\texttt{leaf}\ \ \mathsf{tree}}}{\texttt{node}(\mathbf{3}, \texttt{node}(\mathbf{4}, \texttt{leaf}, \texttt{leaf}), \texttt{leaf})\ \ \mathsf{tree}}\ (\mathbf{3} \in \mathbb{Z})}{\texttt{node}(\mathbf{1}, \texttt{node}(\mathbf{2}, \texttt{leaf}, \texttt{leaf}), \texttt{node}(\mathbf{3}, \texttt{node}(\mathbf{4}, \texttt{leaf}, \texttt{leaf}), \texttt{leaf}))\ \ \mathsf{tree}}\ (\mathbf{1} \in \mathbb{Z})$$

산술식(Arithmetic Expressions)

예: $\mathbf{2}$, $-\mathbf{2}$, $\mathbf{1} + \mathbf{2}$, $\mathbf{1} + (\mathbf{2} * (-\mathbf{3}))$

판단문:

$$e \ \mathsf{exp} \quad \iff \quad e \text{ is an arithmetic expression}$$

추론 규칙:

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

문법:

$$\begin{array}{lrcl} \mathsf{exp} & e & ::= & n \quad (n \in \mathbb{Z}) \\ & & \mid & -e \mid e + e \mid e * e \end{array}$$

증명 예:

$$\dfrac{\dfrac{}{\mathbf{1} \ \mathsf{exp}} \quad \dfrac{\dfrac{}{\mathbf{2} \ \mathsf{exp}} \quad \dfrac{\dfrac{}{\mathbf{3} \ \mathsf{exp}}}{(-\mathbf{3}) \ \mathsf{exp}}}{(\mathbf{2} * (-\mathbf{3})) \ \mathsf{exp}}}{(\mathbf{1} + (\mathbf{2} * (-\mathbf{3}))) \ \mathsf{exp}}$$

명제 논리(Propositional Logic)

논리식(formula) 예:

- T, F
- $T \land F$
- $T \lor F$
- $(T \land F) \land (T \lor F)$
- $T \Rightarrow (F \Rightarrow T)$

명제 논리(Propositional Logic)

문법(Syntax):

$$
\begin{array}{rcl}
A, B & ::= & T \mid F \\
& \mid & \neg A \\
& \mid & A \wedge B \\
& \mid & A \vee B \\
& \mid & A \Rightarrow B
\end{array}
$$

수학적 의미(Denotational Semantics) $[\![A]\!]$:

$$
\begin{array}{rcl}
[\![T]\!] & = & \mathit{true} \\
[\![F]\!] & = & \mathit{false} \\
[\![\neg A]\!] & = & \text{not } [\![A]\!] \\
[\![A_1 \wedge A_2]\!] & = & [\![A_1]\!] \text{ andalso } [\![A_2]\!] \\
[\![A_1 \vee A_2]\!] & = & [\![A_1]\!] \text{ orelse } [\![A_2]\!] \\
[\![A_1 \Rightarrow A_2]\!] & = & [\![A_1]\!] \text{ implies } [\![A_2]\!]
\end{array}
$$

명제 논리(Propositional Logic)

진리표(Truth Table)

$$[\![A_1 \wedge A_2]\!]$$

$[\![A_1]\!]$ \ $[\![A_2]\!]$	*true*	*false*
true	*true*	*false*
false	*false*	*false*

$$[\![A_1 \vee A_2]\!]$$

$[\![A_1]\!]$ \ $[\![A_2]\!]$	*true*	*false*
true	*true*	*true*
false	*true*	*false*

$$[\![A_1 \Rightarrow A_2]\!]$$

$[\![A_1]\!]$ \ $[\![A_2]\!]$	*true*	*false*
true	*true*	*false*
false	*true*	*true*

$$[\![\neg A]\!]$$

$[\![A]\!]$	$[\![\neg A]\!]$
true	*false*
false	*true*

명제 논리(Propositional Logic)

$(T \wedge (T \vee F)) \Rightarrow F$ 의 의미?

$$
\begin{aligned}
& [\![(T \wedge (T \vee F)) \Rightarrow F]\!] \\
& = [\![T \wedge (T \vee F)]\!] \text{ implies } [\![F]\!] \\
& = ([\![T]\!] \text{ andalso } [\![T \vee F]\!]) \text{ implies } \mathit{false} \\
& = (\mathit{true} \text{ andalso } ([\![T]\!] \text{ orelse } [\![F]\!])) \text{ implies } \mathit{false} \\
& = (\mathit{true} \text{ andalso } (\mathit{true} \text{ orelse } \mathit{false})) \text{ implies } \mathit{false} \\
& = (\mathit{true} \text{ andalso } \mathit{true}) \text{ implies } \mathit{false} \\
& = \mathit{true} \text{ implies } \mathit{false} \\
& = \mathit{false}
\end{aligned}
$$

요약

- 귀납법은 컴퓨터 과학 전반에 걸쳐 많이 사용되는 핵심 기법
  - 기본 값(primitive values): 불리안, 문자(character), 정수, 문자열 등
  - 합성 값(compound values): 리스트, 나무 구조, 그래프 등
  - 프로그래밍 언어 문법 및 의미 구조