Programming Languages

Lec 1: Inductive Definitions, Basics of OCaml

김재호

Inductive Definitions

Inductive Definitions

왜?

- 자연수: 1, 2, 3, 4, 5, 6, 7, 8, 9, …
- 어떻게 정의할 수 있을까?
  - 1씩 차이나는 수? 0.1, 1.1, 2.1, …
  - 0보다 큰 수? 2.5, 3.3, …

Inductive Definitions

왜?

- 명확하게 정의하기 위해서는,
    1. 명백한 기준이 되는 지점
    2. 그 기준으로부터 귀납적으로 얻어낼 수 있도록 하는 조건

이 필요하다.

Inductive Definitions

왜?

- 자연수: 1, 2, 3, 4, 5, 6, 7, 8, 9, …
- 자연수는 다음과 같이 정의할 수 있다.
    1. 1은 자연수이다.
    2. 만약 어떤 수에서 1을 뺀 수가 자연수라면, 그 수 역시 자연수이다.
- 이것이 귀납 정의(Inductive Definition)
- 귀납정의에는 3가지 형태가 존재한다.

Inductive Definitions

귀납정의의 3가지 형태

- **Top-down**
  1. 1은 자연수이다.
  2. 만약 어떤 수에서 1을 뺀 수가 자연수라면, 그 수 역시 자연수이다.
- **Bottom-up**
  1. 1은 자연수이다.
  2. 만약 어떤 수가 자연수라면, 그 수에서 1을 더한 수 역시 자연수이다.
- 위 두 가지는 자연스럽지만, PL에서는 조금 더 formal한 형식의 **rule of inference**를 사용

Inductive Definitions

귀납정의의 3가지 형태

- **Rule of Inference**
  - 일반적인 형태: $\frac{A}{B}$
    - 의미: A가 true라면, B가 true이다. (이 때 A를 antecedent, B를 consequent라 함)
  - $\frac{A \quad B}{C}$ 의 형태는? — A와 B가 모두 true라면, C가 true이다.
- 가장 기본적으로 주어지는 기준, 즉 공리(axion)는 $\frac{}{A}$ 와 같이 표현한다.

Inductive Definitions

귀납정의의 3가지 형태

- Rule of Inference로 자연수를 정의한다면?

$$\frac{}{1 \in \mathbb{N}}, \quad \frac{n \in \mathbb{N}}{(n+1) \in \mathbb{N}}$$

- 이 때 3이 자연수임을 알아내기 위해서, 다음과 같이 작성할 수 있다.

(보통은 같은 크기로 작성한다…)

$$\dfrac{\dfrac{\overline{1 \in \mathbb{N}}}{2 \in \mathbb{N}}}{3 \in \mathbb{N}}$$

Rule of Inference

여러가지 예시

- **Strings**

$$\frac{}{\epsilon} \qquad \frac{\alpha}{x\alpha}\ x \in \{\mathtt{a}, \ldots, \mathtt{z}\}$$

- **Lists**

$$\frac{}{\mathsf{nil}} \qquad \frac{l}{n \cdot l}\ n \in \mathbb{Z}$$

- **Binary Trees**

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

Structural Induction
귀납 증명

- 지금까지의 귀납 정의를 활용하면, 이렇게 정의되는 모든 구조에 대해 귀납 증명이 가능!
- 어떤 명제 P(s)에 대해,
  - (Base case) P가 가장 간단한 구조에 대해 참이다
  - (Inductive case) 구조 s의 substructure에 대해 P가 성립하면, P(s) 역시 성립한다
- 위 형태로 증명하는 것을 Structural Induction이라 한다.
- 쉽게 생각해서, 수학적 귀납법의 structural 버전!

앞으로 우리는...

$$\frac{}{\rho, \sigma \vdash n \Rightarrow n, \sigma} \qquad \frac{}{\rho, \sigma \vdash x \Rightarrow \sigma(\rho(x)), \sigma}$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow \mathit{true}, \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow v, \sigma_2}{\rho, \sigma_0 \vdash \texttt{if } E_1 \texttt{ then } E_2 \texttt{ else } E_3 \Rightarrow v, \sigma_2}$$

$$\frac{}{\rho, \sigma \vdash \texttt{proc } x\ E \Rightarrow (x, E, \rho), \sigma} \qquad \frac{\rho, \sigma_0 \vdash E \Rightarrow v, \sigma_1}{\rho, \sigma_0 \vdash x := E \Rightarrow v, [\rho(x) \mapsto v]\sigma_1}$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow v_1, \sigma_1 \qquad [x \mapsto l]\rho, [l \mapsto v_1]\sigma_1 \vdash E_2 \Rightarrow v, \sigma_2}{\rho, \sigma_0 \vdash \texttt{let } x = E_1 \texttt{ in } E_2 \Rightarrow v, \sigma_2}\ l \notin \mathrm{Dom}(\sigma_1)$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow (x, E, \rho'), \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow v, \sigma_2 \qquad [x \mapsto l]\rho', [l \mapsto v]\sigma_2 \vdash E \Rightarrow v', \sigma_3}{\rho, \sigma_0 \vdash E_1\ E_2 \Rightarrow v', \sigma_3}\ l \notin \mathrm{Dom}(\sigma_2)$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow (x, E, \rho'), \sigma_1 \qquad [x \mapsto \rho(y)]\rho', \sigma_1 \vdash E \Rightarrow v', \sigma_2}{\rho, \sigma_0 \vdash E_1\ \langle y \rangle \Rightarrow v', \sigma_2}$$

$$\frac{\rho, \sigma_0 \vdash E_1 \Rightarrow v_1, \sigma_1 \qquad \rho, \sigma_1 \vdash E_2 \Rightarrow v_2, \sigma_2}{\rho, \sigma_0 \vdash E_1; E_2 \Rightarrow v_2, \sigma_2}$$

Basics of OCaml

Functional Programming
함수형 프로그래밍이 뭘까?

- 보통, 함수를 first-class로 취급하는 언어를 함수형 프로그래밍 언어라 한다.
- First-Class란?
  - 변수에 저장될 수 있으며,
  - 함수의 인자로 사용될 수 있고,
  - 함수의 리턴값이 될 수 있는 것
- 쉽게 이야기해, 값처럼 쓰일 수 있는 것을 의미한다! (*엄밀한 정의는 아님)

Functional Programming

함수형 프로그래밍이 뭘까?

- 우리가 수학에서 활용하는 함수는 주어진 입력값에 따라 함수값이 명확하게 정해지는 형태
- 이는 다른 말로 side-effect가 없다고 표현
- 함수형 언어는 대부분의 경우 side-effect가 없도록 함수를 활용한다. (이 말의 정확한 의미는 앞으로 깨닫게 될 것)
- 또한, 함수를 값처럼 활용함에 따라 재귀와 고차함수가 매우 용이하게 쓰인다.
- OCaml은 대표적인 함수형 언어이자, 선언형 언어
- 다만, Haskell과 같은 순수 함수형 언어가 아니기 때문에 여러 특징이 섞여 있음

OCaml Programming

백문이 불어일견

일단 해보자!