4471028: 프로그래밍언어

Lecture 7 — 함수
Functions

임현승
2020 봄학기

$\mathcal{L}^{let}$: 간단한 표현식 언어

문법구조

$$
\begin{array}{llcll}
\text{Program} & P & ::= & E & \\
\text{Expression} & E & ::= & n & \text{정수} \\
 & & \mid & x & \text{변수} \\
 & & \mid & E + E & \text{덧셈식} \\
 & & \mid & E - E & \text{뺄셈식} \\
 & & \mid & \texttt{iszero}\ E & \text{0값 테스트} \\
 & & \mid & \texttt{if}\ E\ \texttt{then}\ E\ \texttt{else}\ E & \text{조건문} \\
 & & \mid & \texttt{let}\ x = E\ \texttt{in}\ E & \text{지역변수 선언}
\end{array}
$$

$\mathcal{L}^{let}$: 간단한 표현식 언어

의미 공간(semantic domain)

$$\begin{array}{rcl} Val & = & \mathbb{Z} + Bool \\ Env & = & Var \to Val \end{array}$$

실행 의미구조(operational semantics)

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

$$\frac{\rho \vdash E_1 \Rightarrow n_1 \qquad \rho \vdash E_2 \Rightarrow n_2}{\rho \vdash E_1 + E_2 \Rightarrow n_1 + n_2} \qquad \frac{\rho \vdash E_1 \Rightarrow n_1 \qquad \rho \vdash E_2 \Rightarrow n_2}{\rho \vdash E_1 - E_2 \Rightarrow n_1 - n_2}$$

$$\frac{\rho \vdash E \Rightarrow 0}{\rho \vdash \texttt{iszero}\ E \Rightarrow true} \qquad \frac{\rho \vdash E \Rightarrow n}{\rho \vdash \texttt{iszero}\ E \Rightarrow false}\ (n \neq 0)$$

$$\frac{\rho \vdash E_1 \Rightarrow true \qquad \rho \vdash E_2 \Rightarrow v}{\rho \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \Rightarrow v} \qquad \frac{\rho \vdash E_1 \Rightarrow false \qquad \rho \vdash E_3 \Rightarrow v}{\rho \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \Rightarrow v}$$

$$\frac{\rho \vdash E_1 \Rightarrow v_1 \qquad [x \mapsto v_1]\rho \vdash E_2 \Rightarrow v}{\rho \vdash \texttt{let}\ x = E_1\ \texttt{in}\ E_2 \Rightarrow v}$$

$\mathcal{L}^{fun}$ = $\mathcal{L}^{let}$ + 함수

$$
\begin{array}{llcll}
\text{Program} & P & ::= & E & \\
\text{Expression} & E & ::= & n & \\
 & & | & x & \\
 & & | & E + E & \\
 & & | & E - E & \\
 & & | & \texttt{iszero}\ E & \\
 & & | & \texttt{if}\ E\ \texttt{then}\ E\ \texttt{else}\ E & \\
 & & | & \texttt{let}\ x = E\ \texttt{in}\ E & \\
 & & | & \texttt{fun}\ x\ \texttt{->}\ E & \text{이름 없는 함수, 람다식}(\lambda x.E) \\
 & & | & E\ E & \text{함수 호출, 함수 적용}
\end{array}
$$

예제

```
let f = fun x -> x - 11
in f (f 77)
```

```
(fun f -> f (f 77)) (fun x -> x - 11)
```

함수의 자유 변수와 묶여있는 변수

- 함수 f의 몸체(body)에서 변수 x가 사용되고 있을 때,
- 만약 x가 f의 형식 인자(formal parameter)라면 x는 함수 f에 "묶여있다(bound)"고 말하며, 이러한 변수를 묶여있는 변수(bound variable)라고 한다.
- 만약 x가 f의 형식 인자가 아니라면 x는 함수 f에서 "자유롭다(free)"고 말하며, 이러한 변수를 자유 변수(free variable)라고 한다.
- 다음 함수에서

```
fun y -> x + y
```

x는 자유 변수이고 y 묶여있는 변수이다.

변수의 정적 유효범위와 동적 유효범위

다음 코드의 결과는 무엇일까?

```
let x = 1 in
let f = fun y -> x + y in
let x = 2 in
f 3
```

함수 `fun y -> x + y`에서 변수 `x`는 자유 변수이다.

함수의 자유 변수의 값을 결정하는 두 가지 방법

- 정적 유효범위(static scoping, lexical scoping): 변수의 유효범위가 프로그램을 컴파일할 때 결정됨. 함수가 호출되면 함수의 몸체를 실행하는데, 이 때 함수가 정의되는 시점에서의 실행 환경을 이용함.
- 동적 유효범위(dynamic scoping): 변수의 유효범위가 프로그램을 실행하는 중에 결정됨. 함수의 몸체가 실행할 때, 그 시점에서의 실행 환경을 이용함.

대부분의 프로그래밍 언어에서는 정적 유효범위를 이용한다.

정적 유효범위를 사용하는 이유

- 동적 유효범위를 사용할 경우 프로그램을 이해하기가 어려움
  - 정적 유효범위를 사용하면 변수 이름이 컴파일 시간에 분석됨(which variable definition each use of a variable refers to)
  - 동적 유효범위를 사용하면 변수 이름이 실행 시간에 분석됨
- 다음 코드의 결과는 무엇일까?

```
let a = 3 in
let f = fun z -> a in
let g = fun a -> f 0 in
let a = 5 in
g 2
```

- 함수에 묶여있는 변수를 해당 변수의 값을 정의하는 식으로 치환하더라도 정적 유효범위에서는 프로그램의 의미가 변하지 않지만, 동적 유효범위에서는 프로그램의 의미가 변할 수 있음

정적 유효범위에서 함수의 실행 의미구조

- 의미 공간(semantic domain):

$$\begin{array}{rcl} Val & = & \mathbb{Z} + Bool + Fun \\ Fun & = & Var \times Exp \times Env \\ Env & = & Var \to Val \end{array}$$

  함수 값(function value)은 함수의 형식 인자, 몸체, 그리고 함수가 정의되는 시점에서의 실행 환경으로 구성된 클로저(closure)

- 의미 추론 규칙:

$$\frac{}{\rho \vdash \texttt{fun}\ x\ \texttt{->}\ E \Rightarrow (x, E, \rho)}$$

$$\frac{\rho \vdash E_1 \Rightarrow (x, E, \rho') \qquad \rho \vdash E_2 \Rightarrow v \qquad [x \mapsto v]\rho' \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}$$

예제

- $clos = (y, x + y, [x \mapsto 1])$
- $\rho_0 = [x \mapsto 1]$
- $\rho_1 = [x \mapsto 1, f \mapsto clos]$
- $\rho_2 = [x \mapsto 2, f \mapsto clos]$
- $\rho_3 = [x \mapsto 1, y \mapsto 3]$

$$
\dfrac{
  \dfrac{}{[] \vdash 1 \Rightarrow 1}
  \quad
  \dfrac{
    \dfrac{}{\rho_0 \vdash \texttt{fun y -> x + y} \Rightarrow clos}
    \quad
    \dfrac{
      \dfrac{}{\rho_1 \vdash 2 \Rightarrow 2}
      \quad
      \dfrac{
        \dfrac{}{\rho_2 \vdash f \Rightarrow clos}
        \quad
        \dfrac{}{\rho_2 \vdash 3 \Rightarrow 3}
        \quad
        \dfrac{\vdots}{\rho_3 \vdash x + y \Rightarrow 4}
      }{\rho_2 \vdash \texttt{f 3} \Rightarrow 4}
    }{\rho_1 \vdash \begin{array}{l}\texttt{let x = 2 in}\\ \texttt{f 3}\end{array} \Rightarrow 4}
  }{\rho_0 \vdash \begin{array}{l}\texttt{let f = fun y -> x + y in}\\ \texttt{let x = 2 in}\\ \texttt{f 3}\end{array} \Rightarrow 4}
}{[] \vdash \begin{array}{l}\texttt{let x = 1 in}\\ \texttt{let f = fun y -> x + y in}\\ \texttt{let x = 2 in}\\ \texttt{f 3}\end{array} \Rightarrow 4}
$$

동적 유효범위

- 의미 공간:

$$\begin{array}{rcl} Val & = & \mathbb{Z} + Bool + Fun \\ Fun & = & Var \times Exp \\ Env & = & Var \to Val \end{array}$$

- 의미 추론 규칙:

$$\frac{}{\rho \vdash \texttt{fun}\ x\ \texttt{->}\ E \Rightarrow (x, E)}$$

$$\frac{\rho \vdash E_1 \Rightarrow (x, E) \qquad \rho \vdash E_2 \Rightarrow v \qquad [x \mapsto v]\rho \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}$$

예제: 동적 유효범위

- $clos = (y, x + y)$
- $\rho_0 = [x \mapsto 1]$
- $\rho_1 = [x \mapsto 1, f \mapsto clos]$
- $\rho_2 = [x \mapsto 2, f \mapsto clos]$
- $\rho_3 = [x \mapsto 2, f \mapsto clos, y \mapsto 3]$

$$
\dfrac{
  \dfrac{}{[\,] \vdash 1 \Rightarrow 1}
  \qquad
  \dfrac{
    \dfrac{}{\rho_0 \vdash \texttt{fun y -> x + y} \Rightarrow clos}
    \qquad
    \dfrac{
      \dfrac{}{\rho_1 \vdash 2 \Rightarrow 2}
      \qquad
      \dfrac{
        \dfrac{}{\rho_2 \vdash f \Rightarrow clos}
        \quad
        \dfrac{}{\rho_2 \vdash 3 \Rightarrow 3}
        \quad
        \dfrac{\vdots}{\rho_3 \vdash x + y \Rightarrow 5}
      }{\rho_2 \vdash \texttt{f 3} \Rightarrow 5}
    }{\rho_1 \vdash \begin{array}{l}\texttt{let x = 2 in}\\ \texttt{f 3}\end{array} \Rightarrow 5}
  }{\rho_0 \vdash \begin{array}{l}\texttt{let f = fun y -> x + y in}\\ \texttt{let x = 2 in}\\ \texttt{f 3}\end{array} \Rightarrow 5}
}{[\,] \vdash \begin{array}{l}\texttt{let x = 1 in}\\ \texttt{let f = fun y -> x + y in}\\ \texttt{let x = 2 in}\\ \texttt{f 3}\end{array} \Rightarrow 5}
$$

커링을 이용하여 여러 개의 인자를 갖는 함수 표현하기

- 인자를 여러 개 있는 함수(multiple argument function)는 함수를 반환하는 고차 함수(higher-order function)을 이용하여 표현 가능

- 예: 두 개의 정수형 인자 x와 y를 받아서 두 수의 합을 반환하는 함수는

  ```
  let f = fun (x, y) -> x + y in
  f (3, 4)
  ```

  인자로 x 하나만 받아서 (나머지 인자 y를 받아서 x와 y의 합을 반환하는) 함수를 반환하는 고차 함수로 표현 가능

  ```
  let f = fun x -> fun y -> x + y in
  f 3 4
  ```

- 여러 개의 인자를 받는 함수를 인자를 하나만 받도록 변환하는 것을 커링(currying)이라고 하며, 인자를 하나만 받도록 변환된 함수를 커리화(curried) 되었다고 한다.

자기 호출 함수

$\mathcal{L}^{fun}$은 자기 호출 함수를 지원하지 않음 (예: 팩토리얼 함수)

```
let f = fun x -> f x in
f 1
```

계산하면:

$$\dfrac{[\, f \mapsto (x, f\ x, [])\,] \vdash f \Rightarrow (x, f\ x, []) \qquad \dfrac{[\, x \mapsto \mathbf{1}\,] \vdash f \Rightarrow\ ? \qquad [\, x \mapsto \mathbf{1}\,] \vdash x \Rightarrow \mathbf{1}}{[\, x \mapsto \mathbf{1}\,] \vdash f\ x \Rightarrow\ ?}}{[\, f \mapsto (x, f\ x, [])\,] \vdash f\ \mathbf{1} \Rightarrow\ ?}$$

$\mathcal{L}^{rec}$: $\mathcal{L}^{fun}$ + 자기 호출 함수

$$
\begin{array}{llll}
\text{Program} & P & ::= & E \\
\text{Expression} & E & ::= & n \\
& & \mid & x \\
& & \mid & E + E \\
& & \mid & E - E \\
& & \mid & \texttt{iszero}\ E \\
& & \mid & \texttt{if}\ E\ \texttt{then}\ E\ \texttt{else}\ E \\
& & \mid & \texttt{let}\ x = E\ \texttt{in}\ E \\
& & \mid & \texttt{let rec}\ f\ x = E\ \texttt{in}\ E \quad \text{자기 호출 함수 선언문} \\
& & \mid & \texttt{fun}\ x\ \texttt{->}\ E \\
& & \mid & E\ E
\end{array}
$$

예제

```
let rec double x =
  if iszero x then 0 else double (x-1) + 2
in
double 6
```

자기 호출 함수의 실행 의미구조

- 의미 공간:

$$
\begin{array}{rcl}
\mathit{Val} & = & \mathbb{Z} + \mathit{Bool} + \mathit{Fun} + \mathit{RecFun} \\
\mathit{Fun} & = & \mathit{Var} \times \mathit{Exp} \times \mathit{Env} \\
\mathit{RecFun} & = & \mathit{Var} \times \mathit{Var} \times \mathit{Exp} \times \mathit{Env} \\
\mathit{Env} & = & \mathit{Var} \to \mathit{Val}
\end{array}
$$

- 의미 추론 규칙:

$$
\frac{[f \mapsto (f, x, E_1, \rho)]\rho \vdash E_2 \Rightarrow v}{\rho \vdash \texttt{let rec}\ f\ x = E_1\ \texttt{in}\ E_2 \Rightarrow v}
$$

$$
\frac{\rho \vdash E_1 \Rightarrow (f, x, E, \rho') \quad \rho \vdash E_2 \Rightarrow v \qquad [x \mapsto v, f \mapsto (f, x, E, \rho')]\rho' \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}
$$

예제

$$
\dfrac{\dfrac{[f \mapsto (f, x, f\ x, [])] \vdash \texttt{f} \Rightarrow (f, x, f\ x, []) \qquad \dfrac{\vdots}{[x \mapsto \mathbf{1}, f \mapsto (f, x, f\ x, [])] \vdash \texttt{f x} \Rightarrow}}{[f \mapsto (f, x, f\ x, [])] \vdash \texttt{f 1} \Rightarrow}}{[] \vdash \texttt{let rec f x = f x in f 1} \Rightarrow}
$$

cf) 동적 유효범위에서의 자기 호출 함수

동적 유효범위 하에서는 자기 호출 함수를 지원하기 위해 실행 의미구조를 확장할 필요가 없음. 아래 프로그램을

```
let f = fun x -> f x in
f 1
```

다음 동적 유효범위 기반 의미구조를 이용하여

$$\frac{\rho \vdash E_1 \Rightarrow (x, E) \qquad \rho \vdash E_2 \Rightarrow v \qquad [x \mapsto v]\rho \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}$$

실행하면

$$\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{[x \mapsto \mathbf{1}, f \mapsto (x, f\ x)] \vdash \texttt{f x} \Rightarrow}}{[x \mapsto \mathbf{1}, f \mapsto (x, f\ x)] \vdash \texttt{f x} \Rightarrow}}{[f \mapsto (x, f\ x)] \vdash \texttt{f 1} \Rightarrow}}{[] \vdash \texttt{let f = fun x -> f x in f 1} \Rightarrow}$$

요약: 문법구조

표현식과 함수로 구성된 튜링 완전한(Turing-complete) 언어:

$$
\begin{array}{rcl}
P & ::= & E \\
E & ::= & n \\
& | & x \\
& | & E + E \\
& | & E - E \\
& | & \texttt{iszero}\ E \\
& | & \texttt{if}\ E\ \texttt{then}\ E\ \texttt{else}\ E \\
& | & \texttt{let}\ x = E\ \texttt{in}\ E \\
& | & \texttt{let rec}\ f\ x = E\ \texttt{in}\ E \\
& | & \texttt{fun}\ x\ \texttt{->}\ E \\
& | & E\ E
\end{array}
$$

요약: 의미구조

$$\frac{}{\rho \vdash n \Rightarrow n} \qquad \frac{}{\rho \vdash x \Rightarrow \rho(x)} \qquad \frac{\rho \vdash E_1 \Rightarrow n_1 \qquad \rho \vdash E_2 \Rightarrow n_2}{\rho \vdash E_1 + E_2 \Rightarrow n_1 + n_2}$$

$$\frac{\rho \vdash E \Rightarrow 0}{\rho \vdash \texttt{iszero}\ E \Rightarrow \mathit{true}} \qquad \frac{\rho \vdash E \Rightarrow n}{\rho \vdash \texttt{iszero}\ E \Rightarrow \mathit{false}}\ (n \neq 0)$$

$$\frac{\rho \vdash E_1 \Rightarrow \mathit{true} \qquad \rho \vdash E_2 \Rightarrow v}{\rho \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \Rightarrow v} \qquad \frac{\rho \vdash E_1 \Rightarrow \mathit{false} \qquad \rho \vdash E_3 \Rightarrow v}{\rho \vdash \texttt{if}\ E_1\ \texttt{then}\ E_2\ \texttt{else}\ E_3 \Rightarrow v}$$

$$\frac{\rho \vdash E_1 \Rightarrow v_1 \qquad [x \mapsto v_1]\rho \vdash E_2 \Rightarrow v}{\rho \vdash \texttt{let}\ x = E_1\ \texttt{in}\ E_2 \Rightarrow v} \qquad \frac{[f \mapsto (f, x, E_1, \rho)]\rho \vdash E_2 \Rightarrow v}{\rho \vdash \texttt{let rec}\ f\ x = E_1\ \texttt{in}\ E_2 \Rightarrow v}$$

$$\frac{}{\rho \vdash \texttt{fun}\ x\ \texttt{->}\ E \Rightarrow (x, E, \rho)}$$

$$\frac{\rho \vdash E_1 \Rightarrow (x, E, \rho') \qquad \rho \vdash E_2 \Rightarrow v \qquad [x \mapsto v]\rho' \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}$$

$$\frac{\rho \vdash E_1 \Rightarrow (f, x, E, \rho') \quad \rho \vdash E_2 \Rightarrow v \quad [x \mapsto v, f \mapsto (f, x, E, \rho')]\rho' \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}$$