SNU 프로그래밍언어 특강

(2.1)

이 광근

kwangkeunyi.snu.ac.kr

여러모양 타입polymorphic type, 다형 타입 계산들

간단한 타입simple type만으로는 담기에 버거운 계산들이 있음

- 타입에 "상관없는" 계산들이 존재
  - 예) 저울, 그네, 내장, 읽기, 붓···
  - 예) identity, list-length, tree-size,···

여러모양 타입polymorphic type, 다형 타입 계산들

예)

저울: $\forall a.a \to \text{real}$

내장: $\forall a \in \text{식음료}.a \to \text{energy}$

읽기: $\forall a \in \text{미디어}.a \to \text{impetus}$

붓: $\forall \alpha \in \text{잉크}.a \to \text{brush}(a)$

list-length: $\forall a.\text{list}(a) \to \text{nat}$

id: $\forall a.a \to a$

저울공장: $\text{nat} \to (\forall a.a \to \text{real})\ \text{list}$

대학: $(\forall a \in \text{미디어}.a \to \text{impetus})$
$\times(\forall a \in \text{잉크}.a \to \text{brush}(a)) \to \text{diffminds}$

a가 어떤 타입이어도 상관없이parametricity 잘작동하는 계산들

계획

- polymorphic type, parametric polymorphism, System F, ad-hoc polymorphism, bounded polymorphism, type class, let-polymorphic types, type checking, type inference

$\forall a.\tau$: 여러모양 타입 polymorphic type, 다형 타입

$$
\begin{array}{rrcll}
\text{간단한 타입} & \tau & \to & \iota & (\text{기본타입:int, bool, } \cdots) \\
 & & | & \tau \to \tau & (\text{함수타입}) \\
\text{타입} & \tau & \to & \iota & \\
 & & | & \tau \to \tau & \\
 & & | & a & (\text{타입변수}) \\
 & & | & \forall a.\tau & (\text{여러모양 타입})
\end{array}
$$

$\forall a.\tau$에서 a의 영역은?

- 간단한 타입들로 제한: "predicative polymorphism"
- 모든 타입들로 확장: "impredicative polymorphism"

여러모양 타입polymorphic type $\forall a.\tau$의 의미

("predicative polymorphism")

직관적($[\![\forall a.\tau]\!] \subseteq$ 프로그램)으로는

$$[\![\forall a.\tau]\!] \stackrel{\text{def}}{=} \bigcap_{t \in \text{SimpleTypes}} [\![\{t/a\}\tau]\!]$$

예)

$$[\![\forall a.a \to a]\!] \stackrel{\text{def}}{=} \bigcap_{t \in \text{SimpleTypes}} [\![t \to t]\!]$$

엄밀히($[\![\forall a.\tau]\!] \subseteq$ 함수)는 부족함

$$[\![\text{nat} \to \text{nat}]\!] \cap [\![\text{bool} \to \text{bool}]\!] = \emptyset$$

여러모양 타입polymorphic type $\forall a.\tau$의 의미

(John Reynolds' model: parametric polymorphism)

$[\![\forall a.\tau]\!] \stackrel{\text{def}}{=}$ a가 어떤 타입이어도 상관없이parametricity
고르게 작동하는uniformly behave 계산들의 모임

예) $[\![\forall a.a \to \mathsf{int}]\!] \stackrel{\text{def}}{=}$

$$\{\{\lambda x_{\mathsf{int}}.1, \lambda x_{\mathsf{bool}}.1, \cdots\}, \{\lambda x_{\mathsf{int}}.2, \lambda x_{\mathsf{bool}}.2, \cdots\}, \cdots$$
$$\{\lambda x_{\mathsf{flesh}}.\mathsf{mass}(x) \times 9, \lambda x_{\mathsf{liquid}}.\mathsf{mass}(x) \times 9, \cdots\}, \cdots\}$$

예) $[\![\forall a.a \to a]\!] \stackrel{\text{def}}{=} \{\{\lambda x_{\mathsf{int}}.x, \lambda x_{\mathsf{bool}}.x, \cdots\}\}$

예) $[\![\forall a.\mathsf{int} \to a]\!] \stackrel{\text{def}}{=} \{\}$ (아직 부족함)

예) $[\![\forall a.\forall b.a \to b]\!] \stackrel{\text{def}}{=} \{\}$ (아직 부족함)

System F (aka Polymorphic Lambda Calculus)

(Girard, Reynolds)

여러모양타입polymorphic type을 만들고 사용하는 프로그래밍 언어

$$
\begin{array}{rrll}
\text{계산식} & E & \to & () \mid x \\
 & & \mid & \lambda x : \tau . E \mid E\ E \\
 & & \mid & \Lambda a . E \qquad \text{(여러모양타입 만들기type abstraction)} \\
 & & \mid & E\ \tau \qquad \text{(여러모양타입 사용하기type application)} \\
\text{타입} & \tau & \to & \iota \\
 & & \mid & \tau \to \tau \\
 & & \mid & a \qquad \text{(타입변수)} \\
 & & \mid & \forall a . \tau \qquad \text{(여러모양타입)}
\end{array}
$$

System F 프로그램 예

- $\lambda f : a \to a.\lambda x : a.f(f\ x) \quad : (a \to a) \to (a \to a)$
- $\Lambda a.\lambda f : a \to a.\lambda x : a.f(f\ x) : \forall a.(a \to a) \to (a \to a)$
- $(\Lambda a.\lambda f : a \to a.\lambda x : a.f(f\ x))\mathsf{int}$
  $\longrightarrow \lambda f : \mathsf{int} \to \mathsf{int}.\lambda x : \mathsf{int}.f(f\ x)$
- $\Lambda a.\lambda f : a \to a.\lambda x : a.x : \forall a.(a \to a) \to (a \to a)$
- $\Lambda a.\lambda f : a \to a.\lambda x : a.f\ x : \forall a.(a \to a) \to (a \to a)$
- $\lambda n : \mathsf{nat}.\Lambda a.\lambda s : a \to a.\lambda z : a.s((n\ a)\ s\ z) : \mathsf{nat} \to \mathsf{nat}$

참고) 자연수/참거짓 표현가능

$$\begin{aligned} \mathsf{nat} &\stackrel{\mathsf{def}}{=} \forall a.(a \to a) \to (a \to a) \\ 1 &\stackrel{\mathsf{def}}{=} \Lambda a.\lambda s : a \to a.\lambda z : a.s\ z \\ \mathsf{succ} &\stackrel{\mathsf{def}}{=} \lambda n : \mathsf{nat}.\Lambda a.\lambda s : a \to a.\lambda z : a.s((n\ a)\ s\ z) \end{aligned}$$

참고) (타입넣고) 재귀함수 표현불가
$Y : \forall a.(a \to a) \to a$ 불가능

System F 실행의미dynamic semantics

적극적계산법call-by-value

$$
\begin{array}{rrcl}
\text{실행문맥} & K & \to & [\,] \\
 & & \mid & K\ E \mid v\ K \\
 & & \mid & K\ \tau \\
\text{값} & v & \to & () \mid \lambda x:\tau.E \mid \Lambda a.E
\end{array}
$$

- 다시 쓸 곳은 다시 쓰면 되고:

$$\frac{E \to E'}{K[E] \to K[E']}$$

- 속에서 어떻게 다시 쓰여지는가 하면:

$$
\begin{array}{rcl}
(\lambda x:\tau.E)\ v & \to & \{v/x\}E \\
(\Lambda a.E)\ \tau & \to & \{\tau/a\}E
\end{array}
$$

System F 실행전 의미$_{\text{static semantics}}$

$\Delta, \Gamma \vdash E : \tau$

타입환경 $\Gamma \in \text{Var} \xrightarrow{\text{fin}} \text{Type}$

묶인타입변수 $\Delta \subseteq \text{TypeVar}$

$$\frac{FTV(\tau) \subseteq \Delta}{\Delta \vdash \tau}$$

$$\frac{}{\Delta, \Gamma \vdash () : \iota} \qquad \frac{\Gamma(x) = \tau}{\Delta, \Gamma \vdash x : \tau}$$

$$\frac{\Delta \vdash \tau_1 \quad \Delta, \Gamma + x : \tau_1 \vdash E : \tau_2 \quad \Delta \vdash \tau_2}{\Delta, \Gamma \vdash \lambda x : \tau_1 . E : \tau_1 \to \tau_2}$$

$$\frac{\Delta, \Gamma \vdash E_1 : \tau \to \tau' \quad \Delta, \Gamma \vdash E_2 : \tau}{\Delta, \Gamma \vdash E_1\ E_2 : \tau'}$$

$$\frac{\Delta \cup \{a\}, \Gamma \vdash E : \tau \quad a \notin \Delta}{\Delta, \Gamma \vdash \Lambda a . E : \forall a . \tau} \qquad \frac{\Delta \vdash \tau \quad \Delta, \Gamma \vdash E : \forall a . \tau'}{\Delta, \Gamma \vdash E\ \tau : \{\tau/a\}\tau'}$$

사실) 실행전의미(타입검사)는 안전함

사실) 타입 유추 알고리즘은 불가능함 ($\nexists \text{Algm}(E) \overset{\text{let}}{=}$ decides if $\exists E', \Gamma, \tau . |E'| = E \ \wedge \ \Gamma \vdash E' : \tau$)

System F의 확장

- $+ \texttt{rec}\ f\ x\ E$
- $+$ 곱타입 $+$ 합타입
- 작은타입$_{\text{subtype}}$, 작은 여러모양 타입$_{\text{subtype polymorphism}}$

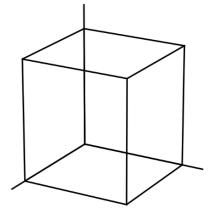
$$\tau <: \tau' \qquad \forall a <: \tau.\tau'$$

- 제한있는 여러모양 타입$_{\text{bounded polymorphism}}$

$$\forall a \in T.\tau$$

(T "type class")

지형도: 람다 큐브lambda cube



프로래밍언어에서 값과 타입을 섞는 8가지

- 원점: 값이 값계산에
  - $(\lambda x.E)\, v \longrightarrow \{v/a\}E$
- x축: 타입이 타입계산에
  - 타입함수type function: 타입으로 타입 만들기

  예) list(int), tree(apple), car(diesel)
- y축: 타입이 값계산에
  - System F
  - $(\Lambda a.E)\, \tau \longrightarrow \{\tau/a\}E$
- z축: 값이 타입계산에
  - 값에기댄 타입dependent type: 타입안에 값계산식

  예) list(3), tree($\lambda x.\bot$),

  $\forall m, n, l.\mathsf{mat}(m, n) \to \mathsf{mat}(n, l) \to \mathsf{mat}(m, l)$