SNU 프로그래밍언어 특강

(1.2)

이 광근

kwangkeunyi.snu.ac.kr

개념/용어 리뷰

- 문법$_{\text{syntax}}$: 생김새, 의미$_{\text{semantics}}$: 속내용
- 집합 정의법: 인덕규칙$_{\text{inductive rules}}$, 인덕$_{\text{induction}}$, 코덕$_{\text{coinduction}}$
- 고정점$_{\text{fixpoint}}$: 최소, 최대
- 무엇 의미구조$_{\text{denotational semantics}}$: CPO, 연속함수
- 실행 의미구조$_{\text{operational semantics}}$: 집합, 증명나무$_{\text{proof tree}}$, 실행발자국$_{\text{transition sequence}}$/실행문맥$_{\text{evaluation context}}$, 실행나무$_{\text{itree}}$
- 딱맞는 의미구조$_{\text{full abstraction semantics}}$
- 증명법: 인덕$_{\text{induction}}$, 고정점인덕$_{\text{fixpoint induction}}$, 고정점 정의 이용법, 겉보기 증명$_{\text{extentional proof}}$

사용한/할 의미개념/의미공간

(CPO/연속함수 vs 집합/유한함수, 구분없이)

- 환경environment

$$\begin{aligned} Env &= Var \to Thing \\ Thing &= Value, \qquad Thing = Loc \end{aligned}$$

- 메모리memory

$$Memory = Loc \to Value$$

- 값value

$$\mathbb{V} = \mathbb{N} + (\mathbb{V} \to \mathbb{V}), \qquad \mathbb{V} = \mathbb{N} + (Exp \times Env)$$

- 마저할일continuation

$$\mathbb{K} = \mathbb{V} \to \mathbb{V}$$

계획

- 값중심언어 확장: 메모리 다루기 + 예외상황 다루기
- 마저할일continuation
- 마저할일드러내기continuation-passing-style transformation
- 할일을 값으로control as value

프로그래밍언어가 제공하는 값의 종류

- 기본값primitives: 숫자, 참거짓, 문자열, ...
- 합성값compounds: 함수, 짝, ...

값 종류마다, 만들기 $\rightleftharpoons$ 사용하기 방법을 제공

- 함수: $\texttt{fn}\, x\, E \rightleftharpoons E\; E$
- 짝: $(E, E) \rightleftharpoons E.\texttt{l},\ E.\texttt{r}$

값중심 언어applicative language

$$
\begin{array}{rrcl}
\text{프로그램식} & E & \to & n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec}\, f\, x\, E \mid E\, E \\
& & \mid & \texttt{let}\, x\, E\, E \mid \texttt{if}\, E\, E\, E \\
& & \mid & (E, E) \mid E.\texttt{l} \mid E.\texttt{r} \mid E + E \mid -E
\end{array}
$$

$$
\begin{array}{rrcl}
\text{실행문맥} & K & \to & [\,] \mid K\, E \mid v\, K \\
\text{값} & v & \to & n \mid \texttt{fn}\, x\, E \mid \texttt{rec}\, f\, x\, E
\end{array}
$$

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

$$
\begin{array}{rcl}
(\texttt{fn}\, x\, E)\, v & \to & \{v/x\}E \\
(\texttt{rec}\, f\, x\, E)\, v & \to & \{(\texttt{rec}\, f\, x\, E)/f, v/x\}E
\end{array}
$$

값중심 언어applicative language 확장

메모리 다루기/메모리주소location를 값으로

$$
\begin{array}{rrcl}
\text{프로그램식} & E & \to & n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec}\, f\, x\, E \mid E\, E \\
 & & \mid & \texttt{ref}\, E \mid E := E \mid\, !E \\
\\
\text{실행문맥} & K & \to & [\,] \mid K\, E \mid v\, K \mid \cdots \\
\text{값} & v & \to & n \mid \texttt{fn}\, x\, E \mid \texttt{rec}\, f\, x\, E \\
 & & \mid & l \\
\text{메모리} & M & \in & Loc \xrightarrow{\text{fin}} Value
\end{array}
$$

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

$$
\begin{array}{rcl}
M, (\texttt{fn}\, x\, E)\, v & \to & M, \{v/x\}E \\
M, (\texttt{rec}\, f\, x\, E)\, v & \to & M, \{(\texttt{rec}\, f\, x\, E)/f, v/x\}E \\
M, l := v & \to & M\{l \mapsto v\}, v \\
\cdots & &
\end{array}
$$

값중심 언어applicative language

예외상황 다루기/튀는 실행흐름control/멀리가기non-local goto

$$
\begin{array}{rrcl}
\text{프로그램식} & E & \to & n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec}\, f\, x\, E \mid E\, E \\
& & \mid & \texttt{raise}\, L \mid E\, \texttt{handle}\, L\, E
\end{array}
$$

$$
\begin{array}{rrcl}
\text{실행문맥} & K & \to & [\,] \mid K\, E \mid v\, K \mid \cdots \\
\text{값} & v & \to & n \mid \texttt{fn}\, x\, E \mid \texttt{rec}\, f\, x\, E \\
\text{예외상황} & \eta & \to & \underline{L}
\end{array}
$$

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

$$
\begin{array}{rcl}
(\texttt{fn}\, x\, E)\, v & \to & \{v/x\}E \\
(\texttt{rec}\, f\, x\, E)\, v & \to & \{(\texttt{rec}\, f\, x\, E)/f, v/x\}E \\
v\, \texttt{handle}\, L\, E & \to & v \\
\texttt{raise}\, L & \to & \underline{L} \\
\underline{L}\, \texttt{handle}\, L\, E & \to & E \\
\cdots & &
\end{array}
$$

마저할일continuation

"방과후 집에오는 길에 놀이방에 들려 동생을 데려오렴"

- 발견: goto-식의 각잡힌 의미를 정의하려다 발견한 개념
- 사용: 프로그래머가 실행흐름을 자유롭게 다루게 하는데 사용하는 개념
- 안경: 번역기compiler가 중간단계에서 하는 변환을 바라보는 안경

마저할일continuation 예

- E_1 + E_2 에서 E_1 계산을 끝내고 마저 할 일은?
- E_1 + E_2 에서 E_2 계산을 끝내고 마저 할 일은?
- E_1 E_2: E_1 계산을 끝내고 마저 할 일은?
- `(goto L)` + E 에서 `goto`를 만나고 마저 할 일은?
- `(raise L)` + E 에서 `raise`를 만나고 마저 할 일은?

마저할일continuation 예

- E_1 + E_2 에서 E_1 계산을 끝내고 마저 할 일은?

  $\lambda x.x$ + E_2
- E_1 + E_2 에서 E_2 계산을 끝내고 마저 할 일은?

- E_1 E_2: E_1 계산을 끝내고 마저 할 일은?

- `(goto L)` + E 에서 `goto`를 만나고 마저 할 일은?

- `(raise L)` + E 에서 `raise`를 만나고 마저 할 일은?

마저할일continuation 예

- E_1 + E_2 에서 E_1 계산을 끝내고 마저 할 일은?

  $\lambda x.x$ + E_2
- E_1 + E_2 에서 E_2 계산을 끝내고 마저 할 일은?

  $\lambda x.v_1$ + x
- E_1 E_2: E_1 계산을 끝내고 마저 할 일은?

- `(goto L)` + E 에서 `goto`를 만나고 마저 할 일은?

- `(raise L)` + E 에서 `raise`를 만나고 마저 할 일은?

마저할일continuation 예

- E_1 + E_2 에서 E_1 계산을 끝내고 마저 할 일은?

  $\lambda x.x$ + E_2
- E_1 + E_2 에서 E_2 계산을 끝내고 마저 할 일은?

  $\lambda x.v_1$ + x
- E_1 E_2: E_1 계산을 끝내고 마저 할 일은?

  $\lambda x.x$ E_2
- `(goto L)` + E 에서 `goto`를 만나고 마저 할 일은?

- `(raise L)` + E 에서 `raise`를 만나고 마저 할 일은?

마저할일continuation 예

- E_1 + E_2 에서 E_1 계산을 끝내고 마저 할 일은?

  $\lambda x.x$ + E_2
- E_1 + E_2 에서 E_2 계산을 끝내고 마저 할 일은?

  $\lambda x.v_1$ + x
- E_1 E_2: E_1 계산을 끝내고 마저 할 일은?

  $\lambda x.x$ E_2
- `(goto L)` + E 에서 `goto`를 만나고 마저 할 일은?

  `L`이름이 붙은 마저할일로 건너뛰기
- `(raise L)` + E 에서 `raise`를 만나고 마저 할 일은?

마저할일continuation 예

- E_1 + E_2 에서 E_1 계산을 끝내고 마저 할 일은?

  $\lambda x.x$ + E_2
- E_1 + E_2 에서 E_2 계산을 끝내고 마저 할 일은?

  $\lambda x.v_1$ + x
- E_1 E_2: E_1 계산을 끝내고 마저 할 일은?

  $\lambda x.x$ E_2
- `(goto L)` + E 에서 `goto`를 만나고 마저 할 일은?

  `L`이름이 붙은 마저할일로 건너뛰기
- `(raise L)` + E 에서 `raise`를 만나고 마저 할 일은?

  `L` 예외상황때 하기로한 마저할일로 건너뛰기

마저할일continuation

- $K[E]$에서 E계산을 마치고(v) 마저 할 일은?

$$\lambda x.K[x]$$

- 즉, 위의 마저할일continuation을 호출하기

$$(\lambda x.K[x])\, v$$

마저할일continuation 사용한 의미구조: 예

$$E \quad \rightarrow \quad \mathtt{1} \mid E + E$$

$$\begin{array}{rcl} [\![E]\!] & \in & (\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \\ [\![\mathtt{1}]\!]\, k & = & k(1) \\ [\![E_1 + E_2]\!]\, k & = & [\![E_1]\!]\, (\lambda n_1.[\![E_2]\!]\, (\lambda n_2.k(n_1 + n_2))) \end{array}$$

- $\forall E, k.k(val(E)) = [\![E]\!]\, k$

마저할일continuation 사용한 의미구조: 값중심 언어

$$E \;\to\; n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec f}\, x\, E \mid E\, E$$

$$\begin{aligned}
[\![E]\!] &\in Env \to \mathbb{K} \to \mathbb{V} \\
\text{환경}\quad \sigma \in Env &= Var_\perp \to \mathbb{V} \\
\text{마저할일}\quad k \in \mathbb{K} &= \mathbb{V} \to \mathbb{V} \\
\text{값}\quad v \in \mathbb{V} &= \mathbb{N}_\perp + (\mathbb{V} \to \mathbb{K} \to \mathbb{V})
\end{aligned}$$

$$\begin{aligned}
[\![n]\!]\,\sigma\,k &= k(n) \\
[\![x]\!]\,\sigma\,k &= k(\sigma(x)) \\
[\![\texttt{fn}\, x\, E]\!]\,\sigma\,k &= k(\lambda v.\lambda k'.[\![E]\!]\,\sigma\{x \mapsto v\}\,k') \\
[\![\texttt{rec}\, f\, x\, E]\!]\,\sigma\,k &= k(lfp(\lambda \bullet .\lambda v.\lambda k'.[\![E]\!]\,\sigma\{x \mapsto v\}\{f \mapsto \bullet\}\,k')) \\
[\![E_1\, E_2]\!]\,\sigma\,k &= [\![E_1]\!]\,\sigma\,(\lambda f.[\![E_2]\!]\,\sigma\,(\lambda v.f\; v\; k))
\end{aligned}$$

마저할일continuation 사용한 의미구조: goto-문

$$
\begin{array}{rcl}
E & \to & n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec f}\, x\, E \mid E\, E \\
 & \mid & L : E \quad L : E \mid \texttt{goto}\, L
\end{array}
$$

$$
\begin{array}{rcl}
[\![E]\!] & \in & Env \to \mathbb{K} \to \mathbb{V} \\
\text{환경} \quad \sigma \in Env & = & (Var + Label)_\perp \to \mathbb{V} \\
\text{마저할일} \quad k \in \mathbb{K} & = & \mathbb{V} \to \mathbb{V} \\
\text{값} \quad v \in \mathbb{V} & = & \mathbb{N}_\perp + (\mathbb{V} \to \mathbb{K} \to \mathbb{V}) + \mathbb{K}
\end{array}
$$

$$
\begin{array}{rcl}
[\![L_1 : E_1 \quad L_2 : E_2]\!]\, \sigma\, k & = & [\![E_1]\!]\, \sigma'\, k_2 \quad \text{이고} \\
 & & \sigma' = \sigma\{L_1 \mapsto k_1\}\{L_2 \mapsto k_2\} \\
 & & k_1 = \lambda v.[\![E_1]\!]\, \sigma'\, k_2 \\
 & & k_2 = \lambda v.[\![E_2]\!]\, \sigma'\, k \\
[\![\texttt{goto}\, L]\!]\, \sigma\, k & = & \sigma(L)(0)
\end{array}
$$

마저할일 드러내기 변환cps transformation: 값중심 언어

마저할일 드러내기continuation-passing-style, cps

$$E \;\to\; n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec}\, \texttt{f}\, x\, E \mid E\, E$$

변환 $\underline{\cdot} \in \mathbb{V}\, Exp \to (\mathbb{K} \to \mathbb{V})\, Exp$

마저할일 $\mathbb{K} = \mathbb{V} \to \mathbb{V}$

$$\begin{aligned}
\underline{n} &= \texttt{fn}\, k\, (k\, n) \\
\underline{x} &= \texttt{fn}\, k\, (k\, x) \\
\underline{\texttt{fn}\, x\, E} &= \texttt{fn}\, k\, (k\, (\texttt{fn}\, x\, \underline{E})) \\
\underline{\texttt{rec}\, f\, x\, E} &= \texttt{fn}\, k\, (k\, (\texttt{rec}\, f\, x\, \underline{E})) \\
\underline{E_1\, E_2} &= \texttt{fn}\, k\, (\underline{E_1}\, (\texttt{fn}\, f\, (\underline{E_2}\, (\texttt{fn}\, v\, (f\, v\, k)))))
\end{aligned}$$

CPS 변환의 올바름

- 무엇 의미구조denotational semantics:

$$[\![E]\!] = [\![\underline{E}]\!]\ id$$

- 실행 의미구조operational semantics:

$$E \xrightarrow{*} v \quad \text{iff} \quad \underline{E}\ \mathsf{id} \xrightarrow{*} v\_$$

$$\begin{aligned} n\_ &= n \\ x\_ &= x \\ (\mathtt{fn}\ x\ E')\_ &= \mathtt{fn}\ x\ \underline{E'} \\ (\mathtt{rec}\ f\ x\ E')\_ &= \mathtt{rec}\ f\ x\ \underline{E'} \end{aligned}$$

- *Call-by-name, call-by-value, and λ-calculus*, Gordon Plotkin, Theoretical Computer Science 1, pp.125–159

CPS변환의 올바름: 겉보기 증명extentional proof

- CPS변환 cpsExp:

$$\texttt{cpsExp} : A\ Exp \ \rightarrow \ ((A \rightarrow A) \rightarrow A)\ Exp$$

- 대응하는 의미세계의 함수 cps:

$$cps : (A \rightarrow A) \rightarrow ((A \rightarrow A) \rightarrow A) \rightarrow ((A \rightarrow A) \rightarrow A)$$

다음 성질을 만족한다(가정):

$$\forall f, f', k.\ (cps\ f)\ f'\ k = f\ (f'\ k)$$

- 다음 겉모습을 증명하자: (의미세계에서 CPS변환의 올바름)

$$\forall F.\ lfpF = (lfp(cps\ F))\ id$$

증명목표:

$$lfp(F) = (lfp(cps\ F))\ id$$

고정점 인덕으로 증명: $P(lfp(F), lfp(cps\ F))$

$$P(f, g) \stackrel{\text{let}}{=} (f = g\ id)$$

$(f \in A, \quad g \in (A \to A) \to A)$

- $\bot_A = \bot_{(A\to A)\to A}(id)$? 네.
- $P(f, g)$ 이면 $P(F\ f, (cps\ F)\ g)$? 즉
  $f = g\ id$ 이면 $F\ f = (cps\ F)\ g\ id$? 네, 왜냐면

$$\begin{array}{rcll} (cps\ F)\ g\ id & = & F\ (g\ id) & (cps \text{ 성질}) \\ & = & F\ f. & (\text{인덕가정}) \end{array}$$

마저할일continuation 사용한 의미구조: 예외상황

$$
\begin{array}{rcl}
E & \to & n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec f}\, x\, E \mid E\, E \\
 & \mid & \texttt{raise}\, L \mid E\, \texttt{handle}\, L\, E
\end{array}
$$

$$
\begin{array}{rcl}
[\![E]\!] & \in & Env \to \mathbb{K} \to \mathbb{H} \to \mathbb{V} \\
\text{환경} \quad \sigma \in Env & = & Var_\perp \to \mathbb{V} \\
\text{마저할일} \quad k \in \mathbb{K} & = & \mathbb{V} \to \mathbb{V} \\
\text{예외처리} \quad h \in \mathbb{H} & = & Exn_\perp \to \mathbb{V} \qquad \text{예외 } L \in Exn \\
\text{값} \quad v \in \mathbb{V} & = & \mathbb{N}_\perp + (\mathbb{V} \to \mathbb{K} \to \mathbb{H} \to \mathbb{V})
\end{array}
$$

$$
\begin{array}{rcl}
[\![\texttt{raise}\, L]\!]\, \sigma\, k\, h & = & h(L) \\
[\![E_1\, \texttt{handle}\, L\, E_2]\!]\, \sigma\, k\, h & = & [\![E_1]\!]\, \sigma\, k\, (h\{L \mapsto [\![E_2]\!]\, \sigma\, k\, h\}) \\
[\![n]\!]\, \sigma\, k\, h & = & k(n) \\
[\![x]\!]\, \sigma\, k\, h & = & k(\sigma(x)) \\
[\![E_1\, E_2]\!]\, \sigma\, k\, h & = & [\![E_1]\!]\, \sigma\, (\lambda f.[\![E_2]\!]\, \sigma\, (\lambda v.f\, v\, k\, h)\, h)\, h \\
\cdots & &
\end{array}
$$

예외상황 녹이기: 마저할일continuation 이용

$$
\begin{array}{rcl}
E & \to & n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec f}\, x\, E \mid E\; E \mid E?\, E : E \\
 & \mid & \texttt{raise}\, L \mid E\, \texttt{handle}\, L\, E
\end{array}
$$

$$
\begin{array}{rrcl}
\text{변환} & \underline{\cdot} & \in & \mathbb{V}\; Exp \to (\mathbb{K} \to \mathbb{H} \to \mathbb{V})\; Exp \\
\text{마저할일} & k \in \mathbb{K} & = & \mathbb{V} \to \mathbb{V} \\
\text{예외처리} & h \in \mathbb{H} & = & Exn \to \mathbb{V} \qquad \text{예외}\; L \in Exn
\end{array}
$$

$$
\begin{array}{rcl}
\underline{\texttt{raise}\, L} & = & \texttt{fn}\, (k, h)\, (h\; L) \\
\underline{E_1\, \texttt{handle}\, L\, E_2} & = & \texttt{fn}\, (k, h)\, \underline{E_1}\, (k, \texttt{fn}\, x\, (x = L?\, (\underline{E_2}\, (k, h)) : h(x))) \\
\underline{n} & = & \texttt{fn}\, (k, h)\, (k\; n) \\
\underline{x} & = & \texttt{fn}\, (k, h)\, (k\; x) \\
\underline{\texttt{fn}\, x\, E} & = & \texttt{fn}\, (k, h)\, (k\, (\texttt{fn}\, x\, \underline{E})) \\
\underline{\texttt{rec}\, f\, x\, E} & = & \texttt{fn}\, (k, h)\, (k\, (\texttt{rec}\, f\; x\, \underline{E})) \\
\underline{E_1\; E_2} & = & \texttt{fn}\, (k, h)\, (\underline{E_1}\, (\texttt{fn}\, f\, (\underline{E_2}\, (\texttt{fn}\, v\, (f\; v\, (k, h)), h)), h)) \\
\cdots & &
\end{array}
$$

마저할일continuation을 값으로control as value

$$
\begin{array}{rcl}
E & \to & n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec f}\, x\, E \mid E\, E \\
 & \mid & \texttt{catch}\, x\, E \mid \texttt{throw}\, x\, E
\end{array}
$$

$$
\begin{array}{rcl}
[\![E]\!] & \in & Env \to \mathbb{K} \to \mathbb{V} \\
\text{환경} \quad \sigma \in Env & = & Var_\perp \to \mathbb{V} \\
\text{마저할일} \quad k \in \mathbb{K} & = & \mathbb{V} \to \mathbb{V} \\
\text{값} \quad v \in \mathbb{V} & = & \mathbb{N}_\perp + (\mathbb{V} \to \mathbb{K} \to \mathbb{V}) + \mathbb{K}
\end{array}
$$

$$
\begin{array}{rcl}
[\![\texttt{catch}\, x\, E]\!]\, \sigma\, k & = & [\![E]\!]\, \sigma\{x \mapsto k\}\, k \\
[\![\texttt{throw}\, x\, E]\!]\, \sigma\, k & = & [\![E]\!]\, \sigma\, (\lambda v.\sigma(x)(v)) \\
[\![\texttt{fn}\, x\, E]\!]\, \sigma\, k & = & k(\lambda v.\lambda k'.[\![E]\!]\, \sigma\{x \mapsto v\}\, k') \\
[\![E_1\, E_2]\!]\, \sigma\, k & = & [\![E_1]\!]\, \sigma\, (\lambda f.[\![E_2]\!]\, \sigma\, (\lambda v.f\ v\ k)) \\
\ldots & &
\end{array}
$$

마저할일continuation을 값으로control as value

$$
\begin{array}{rcl}
E & \to & n \mid x \mid \texttt{fn}\, x\, E \mid \texttt{rec f}\, x\, E \mid E\, E \\
& \mid & \texttt{catch}\, x\, E \mid \texttt{throw}\, x\, E
\end{array}
$$

$$
\begin{array}{rcl}
[\![E]\!] & \in & Env \to \mathbb{K} \to \mathbb{V} \\
\text{환경} \quad \sigma \in Env & = & Var_\perp \to \mathbb{V} \\
\text{마저할일} \quad k \in \mathbb{K} & = & \mathbb{V} \to \mathbb{V} \\
\text{값} \quad v \in \mathbb{V} & = & \mathbb{N}_\perp + (\mathbb{V} \to \mathbb{K} \to \mathbb{V}) + \mathbb{K}
\end{array}
$$

$$
\begin{array}{rcl}
[\![\texttt{catch}\, x\, E]\!]\, \sigma\, k & = & [\![E]\!]\, \sigma\{x \mapsto k\}\, k \\
[\![\texttt{throw}\, x\, E]\!]\, \sigma\, k & = & [\![E]\!]\, \sigma\, (\lambda v.\sigma(x)(v)) \\
[\![\texttt{fn}\, x\, E]\!]\, \sigma\, k & = & k(\lambda v.\lambda k'.[\![E]\!]\, \sigma\{x \mapsto v\}\, k') \\
[\![E_1\, E_2]\!]\, \sigma\, k & = & [\![E_1]\!]\, \sigma\, (\lambda f.[\![E_2]\!]\, \sigma\, (\lambda v.f\ v\ k)) \\
\ldots & &
\end{array}
$$

(C/C++)에서: $\sim \texttt{setjmp}\, x\, E \mid \texttt{longjmp}\, x\, E$

Scheme(Haskell/Scala등)에서: $\sim \texttt{callcc}\, (\texttt{fn}\, x\, E)$

프로그래밍언어가 제공하는 값의 종류

프로래머가 맘대로 다루는as first-class object

- 숫자, 참거짓, 문자열, ...
- 함수, 짝, 메모리주소, 실행순서control, 프로램식, ...

값 종류마다, 만들기 $\rightleftharpoons$ 사용하기 방법을 제공

$$
\begin{array}{rrcl}
\text{함수:} & \texttt{fn}\, x\, E \mid \texttt{rec}\, f\, x\, E & \rightleftharpoons & E\, E \\
\text{짝:} & (E, E) & \rightleftharpoons & E.\texttt{l} \mid E.\texttt{r} \\
\text{메모리주소:} & \texttt{ref}\, E & \rightleftharpoons & E := E \mid\, !E \\
\text{실행순서:} & \texttt{catch}\, x\, E & \rightleftharpoons & \texttt{throw}\, x\, E \\
\text{프로램식:} & \texttt{box}\, E & \rightleftharpoons & \texttt{unbox}\, E \mid \texttt{run}\, E
\end{array}
$$

코드를 값으로expression as value

코드를 계산하는 프로그램

- 일반
  - 메타meta 프로그래밍, 다단계multi-staged 프로그래밍
- 특수
  - 매크로macro, 부분미리계산partial evaluation, 실행중코드만들기run-time code generation
  - 진화프로그래밍
    - 검증불가능한 신경망언어가 아닌 검증가능한 프로그래밍언어의 세계에서
- 메타/다단계를 한 언어안에서

다단계multi-staged 프로그래밍

- 계산이 여러 단계로 나눠짐
- 0단계 코드: 보통의 코드
- $n+1$단계 코드: n단계에서 만들어지는 코드

단계	계산	값
0	보통계산 + 코드만들고실행	보통값 + 코드
> 0	코드끼어넣기	코드

다단계 프로래밍 예 (1/3)

$$
\begin{array}{rcll}
e & ::= & \cdots & \\
 & | & \texttt{box}\, e & \text{코드 만들기} \\
 & | & \texttt{unbox}\, e & \text{코드 끼어넣기} \\
 & | & \texttt{run}\, e & \text{코드 실행하기}
\end{array}
$$

다단계 프로래밍 예 (1/3)

$$
\begin{array}{rcll}
e & ::= & \cdots & \\
 & | & \texttt{box}\, e & \text{코드 만들기} \\
 & | & \texttt{unbox}\, e & \text{코드 끼어넣기} \\
 & | & \texttt{run}\, e & \text{코드 실행하기}
\end{array}
$$

단순 코드 만들기

```
let NULL = box 0
let body = box (if e = unbox(NULL) then abort() ...)
in run body
```

다단계 프로래밍 예 (2/3)

코드 만드는 함수: 코드를 인자로 받아서 코드 만들기

```
let repeatUntil(s,c) =
     box(unbox(s); while unbox(c) do unbox(s))
let xloop = repeatUntil(box(x = x+1), box(x<10))
let x = 0 in run xloop
```

다단계 프로래밍 예 (3/3)

"진화프로그래밍"/특화specialization/부분미리실행partial evaluation

```
power(x,n) = if n=0 then 1 else x * power(x,n-1)
```

v.s. `power(x,3) = x*x*x`

다음과 같이 준비:

```
let power'(n) =
    if n=0 then box(1)
    else box(x * unbox(power' (n-1)))
in run box(fn x (unbox(power' 3)))
```

다단계 프로래밍 실제

- 자유변수가 있는, 열린 코드

```
box(x+1)
```

- 코드끼어넣기때 자유변수가 묶이는 것을 의도하기도

```
box(fn x (unbox(spower 10)))
```

- 코드끼어넣기때 자유변수가 묶이는 것을 피하기도

```
box(fn! x (unbox(spower 10) + x))
```

- 메모리에 자유롭게 보관되는 열린 코드

```
cell := box(x+1); ··· cell := box(y 1);
```

실행발자국 의미구조transitional semantics

$$
\begin{array}{rrcl}
 & \mathit{Exp} \quad e & \to & n \mid x \mid \texttt{fn}\, x\, e \mid \texttt{rec f}\, x\, e \mid e\, e \\
 & & \mid & \texttt{box}\, e \mid \texttt{unbox}\, e \mid \texttt{run}\, e \\
 & & & \\
 & \mathit{Value}^0 \quad v^0 & \to & n \mid \texttt{fn}\, x\, E \mid \texttt{rec}\, f\, x\, E \mid \texttt{box}\, v^1 \\
(n > 0) & \mathit{Value}^n \quad v^n & \to & n \mid x \mid \texttt{fn}\, x\, v^n \mid v^n\, v^n \mid \texttt{rec}\, f\, x\, v^n \\
 & & \mid & \texttt{box}\, v^{n+1} \mid \texttt{unbox}\, v^{n-1} \mid \texttt{run}\, v^n
\end{array}
$$

(APP) $$\frac{e_1 \xrightarrow{n} e_1'}{e_1\ e_2 \xrightarrow{n} e_1'\ e_2} \qquad \frac{e \xrightarrow{n} e' \quad v \in Value^n}{v\ e \xrightarrow{n} v\ e'}$$

$$(\mathtt{fn}\ x\ e)\ v \xrightarrow{0} \{x \overset{0}{\mapsto} v\}e$$

$$(\mathtt{rec}\ f\ x\ e)\ v \xrightarrow{0} \{x \overset{0}{\mapsto} v, f \overset{0}{\mapsto} \mathtt{rec}\ f\ x\ e\}e$$

(BOX) $$\frac{e \xrightarrow{n+1} e'}{\mathtt{box}\ e \xrightarrow{n} \mathtt{box}\ e'}$$

(RUN) $$\frac{e \xrightarrow{n} e'}{\mathtt{run}\ e \xrightarrow{n} \mathtt{run}\ e'} \qquad \frac{v \in Value^1 \quad FV_0(v) = \varnothing}{\mathtt{run}\ (\mathtt{box}\ v) \xrightarrow{0} v}$$

(UNB) $$\frac{e \xrightarrow{n} e'}{\texttt{unbox}\, e \xrightarrow{n+1} \texttt{unbox}\, e'} \qquad \frac{v \in Value^1}{\texttt{unbox}\, (\texttt{box}\, v) \xrightarrow{1} v}$$

(ABS) $$\frac{e \xrightarrow{n+1} e'}{\texttt{fn}\, x\, e \xrightarrow{n+1} \texttt{fn}\, x\, e'}$$

(FIX) $$\frac{e \xrightarrow{n+1} e'}{\texttt{rec}\, f\, x\, e \xrightarrow{n+1} \texttt{rec}\, f\, x\, e'}$$

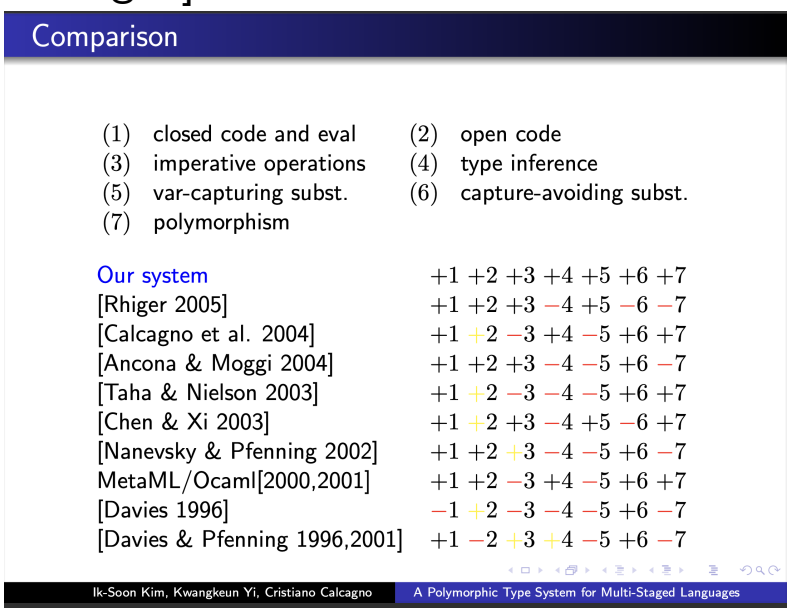
다단계 프로그래밍언어 연구

정신없는 다단계. 편안한 다단계 프로그래밍을 위하여:

다단계 프로그래밍언어 연구

정신없는 다단계. 편안한 다단계 프로그래밍을 위하여:

- 자동검산이 필요
  - 정적타입시스템static type system: "종결자" [POPL'06 Kim, Yi, Calcagno]



Comparison

(1) closed code and eval
(2) open code
(3) imperative operations
(4) type inference
(5) var-capturing subst.
(6) capture-avoiding subst.
(7) polymorphism

Our system	+1 +2 +3 +4 +5 +6 +7
[Rhiger 2005]	+1 +2 +3 −4 +5 −6 −7
[Calcagno et al. 2004]	+1 +2 −3 +4 −5 +6 +7
[Ancona & Moggi 2004]	+1 +2 +3 −4 −5 +6 −7
[Taha & Nielson 2003]	+1 +2 −3 −4 −5 +6 +7
[Chen & Xi 2003]	+1 +2 +3 −4 +5 −6 +7
[Nanevsky & Pfenning 2002]	+1 +2 +3 −4 −5 +6 −7
MetaML/Ocaml[2000,2001]	+1 +2 −3 +4 −5 +6 +7
[Davies 1996]	−1 +2 −3 −4 −5 +6 −7
[Davies & Pfenning 1996,2001]	+1 −2 +3 +4 −5 +6 −7

Ik-Soon Kim, Kwangkeun Yi, Cristiano Calcagno A Polymorphic Type System for Multi-Staged Languages

다단계 프로그래밍언어 연구

다단계 프로그래밍언어 연구

- 일반적인 정적분석은 어떻게?
  - 다단계 녹이기 + 정적분석 + 분석결과 재구성 [POPL'11 Choi, Aktemur, Yi, Tatsuda]
  - 다른 방식도 가능?

다단계 프로그래밍언어 연구

- 일반적인 정적분석은 어떻게?
  - 다단계 녹이기 + 정적분석 + 분석결과 재구성 [POPL'11 Choi, Aktemur, Yi, Tatsuda]
  - 다른 방식도 가능?
- 검증을 위한 프로그램논리program logic는?

다단계 프로그래밍언어 연구

- 일반적인 정적분석은 어떻게?
  - 다단계 녹이기 + 정적분석 + 분석결과 재구성 [POPL'11 Choi, Aktemur, Yi, Tatsuda]
  - 다른 방식도 가능?
- 검증을 위한 프로그램논리$_{\text{program logic}}$는?
- 진화프로그래밍의 두 기둥을 언어에 담기:
  - "system I" 신경망/기계학습 기둥: 상위논리로 커버못하는 문제풀이 진화프로그래밍
  - "system II" 다단계 프로그래밍 기둥: 상위논리로 커버하는 문제풀이 진화프로그래밍
  - 예) (specialize power x) ∘ (learn dnn x)