SNU 프로그래밍언어 특강

(1.1)

이 광근

kwangkeunyi.snu.ac.kr

계획

실행 의미구조operational semantics

- 큰보폭으로
  - 구조물로structural operational semantics
  - 실행 = 증명 나무
  - 인덕규칙induction rule, 추론/증명규칙inference/proof rule
- 작은보폭으로
  - 발자국으로transitional semantics
  - 실행 = 발자국 줄
  - 실행발자국transition sequence, 다시쓰기rewriting, 실행문맥evaluation context

실행 의미구조operational semantics

프로그램 실행과정을 드러내는 정의.

- 충분히 엄밀
- 조립식compositional이 아닐 수 있다
- 하지만 인덕방식inductive이다
  - 프로그램 구조를 따라 인덕
  - 프로그램 이외의 것을 따라 인덕

의미공간semantic domain

보통의 집합, 꼭 CPO일 필요는 없슴

- 인덕, 원소나열법, 조건제시번, $S + T, S \times T, S \overset{\text{fin}}{\to} T$
- 일반 함수집합 만들기 $\to$를 사용하면 곤란

스타일1: 프로그램의 의미 = 증명

호칭: 큰보폭으로$_{\text{big-step semantics}}$,
구조물로$_{\text{structural operational semantics}}$, 자연스레$_{\text{natural semantics}}$,
관계로$_{\text{relational semantics}}$

- 증명 규칙$_{\text{inference rule}}$: 대개 프로그램식 생김새마다 하나이상씩

$$\frac{\cdots}{(M, x := E, M')} \qquad \frac{\cdots}{(M, C_1 \,;\, C_2, M')}$$

$$\frac{\cdots}{(M, \texttt{if}\ E\ C_1\ C_2, M')} \qquad \frac{\cdots}{(M, \texttt{while}\ E\ C, M')}$$

$$\frac{\cdots}{(M, E, M')}\ (\text{“structural rule”})$$

- 증명나무들의 집합: 인덕으로$_{\text{inductive}}$ 정의
- 유한실행과정만 $\Leftarrow$ 인덕$_{\text{induction}}$, 무한실행과정까지 $\Leftarrow$ 코덕$_{\text{coinduction}}$

예: 명령형 언어

명령문 C와 정수식 e의 의미는 (메모리 M 에서)

$$M \vdash C \Rightarrow M' \quad \text{와} \quad M \vdash e \Rightarrow v$$

의 유한한 증명나무

- 증명 불가능 $\Leftrightarrow$ C는 메모리 M에서 무의미

$$\begin{array}{rcrcl} M & \in & Memory & = & Var \xrightarrow{\text{fin}} Val \\ v & \in & Val & = & \mathbb{Z} \end{array}$$

$$\frac{}{M \vdash \texttt{skip} \Rightarrow M}$$

$$\frac{M \vdash E \Rightarrow v}{M \vdash x \texttt{ := } E \Rightarrow M\{x \mapsto v\}}$$

$$\frac{M \vdash C_1 \Rightarrow M_1 \quad M_1 \vdash C_2 \Rightarrow M_2}{M \vdash C_1 \texttt{ ; } C_2 \Rightarrow M_2}$$

$$\frac{M \vdash E \Rightarrow 0 \quad M \vdash C_2 \Rightarrow M'}{M \vdash \texttt{if}\ E\ C_1\ C_2 \Rightarrow M'}$$

$$\frac{M \vdash E \Rightarrow v \quad M \vdash C_1 \Rightarrow M'}{M \vdash \texttt{if}\ E\ C_1\ C_2 \Rightarrow M'}\ v \neq 0$$

$$\frac{M \vdash E \Rightarrow 0}{M \vdash \texttt{while}\ E\ C \Rightarrow M}$$

$$\frac{M \vdash E \Rightarrow v \quad M \vdash C \Rightarrow M_1 \quad M_1 \vdash \texttt{while}\ E\ C \Rightarrow M_2}{M \vdash \texttt{while}\ E\ C \Rightarrow M_2}\ v \neq 0$$

$$\frac{}{M \vdash n \Rightarrow n}$$

$$\frac{}{M \vdash x \Rightarrow M(x)}$$

$$\frac{M \vdash E_1 \Rightarrow v_1 \quad M \vdash E_2 \Rightarrow v_2}{M \vdash E_1 \texttt{ + } E_2 \Rightarrow v_1 + v_2}$$

$$\frac{M \vdash E \Rightarrow v}{M \vdash \texttt{-}\, E \Rightarrow -v}$$

$$C \stackrel{\mathsf{let}}{=} x := 1 \ ; \ y := x + 1$$

$$\dfrac{\dfrac{}{\emptyset \vdash 1 \Rightarrow 1}}{\emptyset \vdash x := 1 \Rightarrow \{x \mapsto 1\}} \qquad \dfrac{\dfrac{\{x \mapsto 1\} \vdash x \Rightarrow 1 \qquad \{x \mapsto 1\} \vdash 1 \Rightarrow 1}{\{x \mapsto 1\} \vdash x + 1 \Rightarrow 2}}{\{x \mapsto 1\} \vdash y := x + 1 \Rightarrow \{x \mapsto 1, y \mapsto 2\}}$$

$$\emptyset \vdash C \Rightarrow \{x \mapsto 1, y \mapsto 2\}$$

예: 값중심 언어, 적극적계산법call-by-value

프로그램식	E	$\rightarrow$ n	자연수
		$\mid$ x	변수
		$\mid$ $\mathtt{fn}\, x\, E$	함수값
		$\mid$ $\mathtt{rec}\, x\, E$	재귀값
		$\mid$ $E\, E$	함수적용

식 E의 의미:

$$\sigma \vdash E \Rightarrow v$$

의 유한한 증명나무(유한 실행과정)

- 증명 불가능 $\Leftrightarrow$ E는 환경 σ에서 무의미

$$
\begin{array}{rrcl}
\text{값} & v \in \mathbb{V} & = & \mathbb{N} + \mathbb{C} \\
\text{함수값} & \mathbb{C} & = & Exp \times Env \\
\text{환경} & \sigma \in Env & = & Var \xrightarrow{\text{fin}} \mathbb{V}
\end{array}
$$

$$
\frac{}{\sigma \vdash n \Rightarrow n}
$$

$$
\frac{}{\sigma \vdash x \Rightarrow \sigma(x)}
$$

$$
\frac{}{\sigma \vdash \texttt{fn}\, x\, E \Rightarrow (\texttt{fn}\, x\, E, \sigma)}
$$

$$
\frac{\sigma \vdash E_1 \Rightarrow (\texttt{fn}\, x\, E, \sigma') \quad \sigma \vdash E_2 \Rightarrow v \quad \sigma'\{x \mapsto v\} \vdash E \Rightarrow v'}{\sigma \vdash E_1\, E_2 \Rightarrow v'}
$$

$\texttt{rec}\, x\, E$ 의미: 안1

$$\frac{\sigma\{x \mapsto v\} \vdash E \Rightarrow v}{\sigma \vdash \texttt{rec}\, x\, E \Rightarrow v}$$

- 재귀함수의 v가 존재? $\mathbb{C}$을 확장해야:

$$\begin{aligned}
\mathbb{V} &= \mathbb{N} + \mathbb{C} \\
\mathbb{C} &= Expr \times Env + Expr \times RecEnv \\
\sigma \in Env &= Var \xrightarrow{\text{fin}} \mathbb{V} \\
\alpha, \mu\alpha.\sigma \in RecEnv &= EnvVar + EnvVar \times Env
\end{aligned}$$

- 생김새 다른 값을 같게 여기기 $v \equiv v'$ 정의해야:

$v \equiv v'$의 핵

- $\mu\alpha.\sigma$는 "한꺼풀 베껴서 바꿔친것"과 같은것으로 한다:

$$\mu\alpha.\sigma \equiv \{\mu\alpha.\sigma/\alpha\}\sigma$$

(α는 모두 다름)

- 바꿔치기는 구조를 유지homomorphic하며 (늘)

$$\begin{array}{lcl} \{\mu\alpha.\sigma/\alpha\}\sigma' & \stackrel{\text{def}}{=} & \bigcup_{(x\mapsto v)\in\sigma'}\{x \mapsto \{\mu\alpha.\sigma/\alpha\}v\} \\ \{\mu\alpha.\sigma/\alpha\}v & \stackrel{\text{def}}{=} & \cdots \end{array}$$

- $v \equiv v'$와 $\sigma \equiv \sigma'$은 구조를 따라 인덕inductive으로 (늘)

그래서

- $\texttt{rec}\ f\ (\texttt{fn}\ x\ E)$의 의미는:

$$\frac{\sigma\{f \mapsto v\} \vdash \texttt{fn}\ x\ E \Rightarrow v}{\sigma \vdash \texttt{rec}\ f\ (\texttt{fn}\ x\ E) \Rightarrow v}$$

여기서

$$v \stackrel{\text{let}}{=} (\texttt{fn}\ x\ E, \mu\alpha.\sigma\{f \mapsto (\texttt{fn}\ x\ E, \alpha)\})$$

- $\texttt{rec}\ x\ (x\texttt{+1})$ 의 의미는?

$\texttt{rec}\ x\ E$ 의미: 안2

- 의미공간은 그대로 두고, 재귀는 함수로만 제한하고:

$$E \quad \to \quad \cdots \mid \texttt{fn}\ x\ E \mid \texttt{rec}\ f\ x\ E \mid E\ E$$

$$\frac{}{\sigma \vdash \texttt{rec}\ f\ \ x\ E \Rightarrow (\texttt{rec}\ f\ x\ E, \sigma)}$$

$$\frac{\sigma \vdash E_1 \Rightarrow (\texttt{rec}\ f\ x\ E, \sigma') \qquad \sigma \vdash E_2 \Rightarrow v_2 \qquad \sigma'\{f \mapsto (\texttt{rec}\ f\ x\ E, \sigma'), x \mapsto v_2\} \vdash E \Rightarrow v}{\sigma \vdash E_1\ E_2 \Rightarrow v}$$

스타일2: 프로그램의 의미 = 실행발자국 (transition sequence)

작은 보폭으로$_{\text{small-step semantics}}$·
실행발자국 의미구조$_{\text{transitional semantics}}$

- 프로그램의 실행 = 무언가가 변해간 발자국
- 무한한 실행과정? 무한히 늘어선 발자국, 무한히 변해가기
- 예) 1+2+3의 의미 = 1+2+3 $\rightarrow$ 3+3 $\rightarrow$ 6

- 실행발자국 규칙transition rule

$$\frac{\cdots}{(M, C) \to (M', C')}$$

- 이면, 프로그램 C의 실행발자국은 증명들의 일렬:

$$\frac{\cdots}{(M_0, C) \to (M_1, C_1)} \quad \Rightarrow \quad \frac{\cdots}{(M_1, C_1) \to (M_2, C_2)} \quad \Rightarrow \cdots$$

명령문 C와 정수식 e의 의미는 (메모리 M에서), 실행발자국

$$\begin{array}{rcl} (M,C) & \to & (M_1,C_1) \\ (M_1,C_1) & \to & (M_2,C_2) \\ & \vdots & \\ (M_n,C_n) & \to & (M',\mathsf{done}) \end{array}$$

와

$$\begin{array}{rcl} (M_0,e) & \to & (M_1,e_1) \\ (M_1,e_1) & \to & (M_2,e_2) \\ & \vdots & \\ (M_m,e_m) & \to & (M'',v) \end{array}$$

꼴을 증명한 나무들의 일렬(실행 발자국)

예: 명령형 언어

$$
\begin{array}{rcrcl}
M & \in & Memory & = & Var \xrightarrow{\text{fin}} Val \\
v & \in & Val & = & \mathbb{Z} \\
C & \to & \cdots & | & \mathsf{done}
\end{array}
$$

$$\frac{}{(M, \texttt{skip}) \to (M, \mathsf{done})}$$

$$\frac{(M, E) \to (M, E')}{(M, x := E) \to (M, x := E')}$$

$$\frac{}{(M, x := v) \to (M\{x \mapsto v\}, \mathsf{done})}$$

$$\frac{(M, C_1) \to (M', C_1')}{(M, C_1 \ ; \ C_2) \to (M', C_1' \ ; \ C_2)}$$

$$\frac{}{(M, \mathsf{done} \ ; \ C_2) \to (M, C_2)}$$

$$\frac{(M, E) \to (M, E')}{(M, \texttt{if}\ E\ C_1\ C_2) \to (M, \texttt{if}\ E'\ C_1\ C_2)}$$

$$\frac{}{(M, \texttt{if}\ 0\ C_1\ C_2) \to (M, C_1)}$$

$$\frac{}{(M, \texttt{if}\ n\ C_1\ C_2) \to (M, C_1)}\ n \neq 0$$

$$\frac{}{(M, \texttt{while}\ E\ C) \to (M, \texttt{if}\ E\ C\ \texttt{;}\ \texttt{while}\ E\ C\ \texttt{skip})}$$

$$\frac{}{(M, x) \to (M, M(x))}$$

$$\frac{(M, E_1) \to (M, E_1')}{(M, E_1 \texttt{ + } E_2) \to (M, E_1' \texttt{ + } E_2)}$$

$$\frac{(M, E_2) \to (M, E_2')}{(M, v_1 \texttt{ + } E_2) \to (M, v_1 \texttt{ + } E_2')}$$

$$\frac{}{(M, v_1 \texttt{ + } v_2) \to (M, v_1 + v_2)}$$

$$\underbrace{x := 1}_{C_1} \;;\; \underbrace{y := x + 1}_{C_2}$$

$$\frac{}{(\emptyset, x := 1 \;;\; C_2) \to (\{x \mapsto 1\}, \mathsf{done} \;;\; C_2)}$$

$$\frac{}{(\{x \mapsto 1\}, \mathsf{done} \;;\; C_2) \to (\{x \mapsto 1\}, C_2)}$$

$$\frac{\dfrac{}{(\{x \mapsto 1\}, x + 1) \to (\{x \mapsto 1\}, 1 + 1)}}{(\{x \mapsto 1\}, C_2) \to (\{x \mapsto 1\}, y := 1 + 1)}$$

$$\frac{\dfrac{}{(\{x \mapsto 1\}, 1 + 1) \to (\{x \mapsto 1\}, 2)}}{(\{x \mapsto 1\}, y := 1 + 1) \to (\{x \mapsto 1\}, y := 2)}$$

$$\frac{}{(\{x \mapsto 1\}, y := 2) \to (\{x \mapsto 1, y \mapsto 2\}, \mathsf{done})}$$

문맥구조를 통해서 evaluation-context semantics

실행발자국 의미구조를 표현하는 한 방식

- 실행발자국 규칙 = 프로그램 다시 쓰기
- 어디를 다시 써(계산 해)?
  실행문맥 evaluation context 이 결정
- 무엇으로 다시 써?
  다시쓰기 규칙 rewriting rule 이 결정

실행문맥$_{\text{evaluation context}}$ K = 다시 쓸 부분이 정의되어 있는 프로그램

- K가 문법적으로 정의 가능
- 다시 쓸 부분이 $[\,]$(빈칸)으로 표현됨
- 다시 쓸 부분 $[\,]$을 품은 프로그램을 "$K[\,]$"로 표현
- 다시 쓸 부분이 E인 프로그램은 "$K[E]$"로 표현

예: 명령형 언어

$$
\begin{array}{rcl}
C & \to & \texttt{skip} \mid x := E \mid C \,;\, C \\
 & \mid & \texttt{if}\ E\ C\ C \mid \texttt{while}\ E\ C \\
E & \to & n \mid x \mid E + E \mid - E
\end{array}
$$

$$
\begin{array}{rcl}
\text{실행문맥}\ K & \to & [\,] \\
 & \mid & x := K \\
 & \mid & K \,;\, C \\
 & \mid & r \,;\, K \\
 & \mid & \texttt{if}\ K\ C\ C \\
 & \mid & \texttt{while}\ K\ C \\
 & \mid & K + E \\
 & \mid & r + K \\
 & \mid & - K \\
\text{결과}\ r & \to & n \mid \text{done}
\end{array}
$$

- 다시 쓸 곳은 다시 쓰면 되고:

$$\frac{(M, C) \to (M', C')}{(M, K[C]) \to (M', K[C'])}$$

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

- 속에서 어떻게 다시 쓰여지는가 하면:

$$\begin{aligned}
(M, x := v) &\to (M\{x \mapsto v\}, \mathsf{done}) \\
(M, \mathsf{done}\ ;\ \mathsf{done}) &\to (M, \mathsf{done}) \\
(M, \mathtt{if}\ 0\ C_1\ C_2) &\to (M, C_1) \\
(M, \mathtt{if}\ v\ C_1\ C_2) &\to (M, C_2) \qquad (v \neq 0) \\
(M, \mathtt{while}\ 0\ C) &\to (M, \mathsf{done}) \\
(M, \mathtt{while}\ v\ C) &\to (M, C\ ;\ \mathtt{while}\ E\ C) \qquad (v \neq 0) \\
(M, v_1 + v_2) &\to (M, v) \qquad (v = v_1 + v_2) \\
(M, -v) &\to (M, -v) \\
(M, x) &\to (M, M(x))
\end{aligned}$$

$x := 1 \ ; \ y := x + 1$ 의 의미:

$$\frac{(\emptyset, x := 1) \to (\{x \mapsto 1\}, \mathsf{done})}{(\emptyset, [x := 1] \ ; \ y := x + 1) \to (\{x \mapsto 1\}, \mathsf{done} \ ; \ y := x + 1)}$$

다음은,

$$\frac{(\{x \mapsto 1\}, x) \to (\{x \mapsto 1\}, 1)}{(\{x \mapsto 1\}, \mathsf{done} \ ; \ y := [x] + 1) \to (\{x \mapsto 1\}, \mathsf{done} \ ; \ y := 1 + 1)}$$

다음은,

다음은,

$$\frac{(\{x \mapsto 1\}, y := 2) \to (\{x \mapsto 1, y \mapsto 2\}, \mathsf{done})}{(\{x \mapsto 1\}, \mathsf{done} \ ; \ [y := 2]) \to (\{x \mapsto 1, y \mapsto 2\}, \mathsf{done} \ ; \ \mathsf{done})}$$

다음은,

$$(\{x \mapsto 1, y \mapsto 2\}, \mathsf{done} \ ; \ \mathsf{done}) \to (\{x \mapsto 1, y \mapsto 2\}, \mathsf{done}).$$

예: 값중심 언어, 적극적계산법call-by-value

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

실행문맥 $K \to [\,] \mid K\, E \mid v\, K$

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

- 다시 쓸 곳은 다시 쓰면 되고:

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

- 속에서 어떻게 다시 쓰여지는가 하면:

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

바꿔치기$_{\text{substitution}}$ 정의

- 바꿔치기$_{\text{substitution}}$: 변수 바꿔치기 규칙들

$$\{t_1/x_1, \cdots, t_k/x_k\},\ S \quad \in \quad 2^{Var \times Thing}$$

x_i들은 모두 다름

- $S\square \stackrel{\text{let}}{=}$ $\square$를 S바꿔치기한 결과, $S_2 S_1 \square \stackrel{\text{let}}{=} S_2(S_1 \square)$

위에서, 식E를 S바꿔치기한 결과는 E구조를 유지homomorphic (늘):

$$
\begin{aligned}
Sn &= n \\
Sx &= \begin{cases} t & \text{if } t/x \in S \\ x & \text{if } t/x \notin S \end{cases} \\
S(\texttt{fn}\, x\, E) &= \texttt{fn}\, x\, (SE) \quad (x \notin S) \\
S(\texttt{rec}\, f\, x\, E) &= \texttt{rec}\, f\, x\, (SE) \quad (x, f \notin S) \\
S(E_1 E_2) &= (SE_1)(SE_2)
\end{aligned}
$$

언어 확장1

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

- 다시 쓸 곳은 다시 쓰면 되고:

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

- 속에서 어떻게 다시 쓰여지는가 하면:

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

실행의미 예

- `1+(2+3)`
- `(fn x (x+1)) 2`
- `(fn x (x 1))(fn y (y+2))`
- `let x 1 ((fn y (x+y)) 2)`
- `(rec f x (if x 1 (f (x+1)))) 2`

설탕구조syntactic sugar

이것만으로 완전Turing-complete:

- x, $\texttt{fn}\, x\, E$, $E\ E$

참고) 설탕구조는: (적극적계산법call-by-value)

- 기본값: 참거짓, 부울연산, $\texttt{if}\, E\, E\, E$, 자연수, 자연수연산
- 구조물: (E, E), $E\texttt{.l}$, $E\texttt{.r}$
- $\texttt{rec}\, f\, x\, E$
- $\texttt{let}\, x\, E\, E$