

SNU 4541.664A Program Analysis

Note 3

Prof. Kwangkeun Yi

Seoul National University

과정을 드러내는 의미구조(*operational semantics*)

프로그램 실행 과정을 정의.

- ▶ 충분히 엄밀
- ▶ 조립식(compositional)이 아닐 수 있다
- ▶ 하지만 귀납적(inductive)이다
 - ▶ 프로그램 구조를 따라 되돌기도
 - ▶ 프로그램 이외의 것을 따라 되돌기도

의미공간(*semantic domain*)

보통의 집합, 꼭 CPO일 필요는 없음

- ▶ 귀납법, 원소나열법, 조건제시법, $S \cup T, S + T, S \times T, S \xrightarrow{\text{fin}} T$

$$S \xrightarrow{\text{fin}} T = \{f \mid f \in S' \rightarrow T, S' \subseteq^{\text{fin}} S\}$$

- ▶ 일반 함수집합 만들기 \rightarrow 를 사용하면 곤란

Style 1: 프로그램의 의미 = 증명

structural operational semantics,
natural semantics, relational semantics

- ▶ 증명 규칙(*inference rule*): 프로그램 구조마다 하나 이상 씩

$$\frac{\dots}{(M, \text{skip}, M')}$$

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

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

$$\frac{\dots}{(M, \text{if } E \text{ } C_1 \text{ } C_2, M')}$$

$$\frac{\dots}{(M, \text{while } E \text{ } C, M')}$$

- ▶ 무한한 실행과정? 무한한 증명

예

명령문 C 와 정수식 e 의 의미는 (메모리 M 에서)

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

의 증명

- ▶ 증명 불가능 $\Leftrightarrow C$ 는 메모리 M 에서 무의미

큰 보폭으로(*big-step operational semantics*)

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

$$\overline{M \vdash \text{skip} \Rightarrow M}$$

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\frac{\emptyset \vdash 1 \Rightarrow 1}{\emptyset \vdash x := 1 \Rightarrow \{x \mapsto 1\}} \quad \frac{\frac{\{x \mapsto 1\} \vdash x \Rightarrow 1 \quad \{x \mapsto 1\} \vdash 1 \Rightarrow 1}{\{x \mapsto 1\} \vdash x + 1 \Rightarrow 2}}{\{x \mapsto 1\} \vdash y := x + 1 \Rightarrow \{x \mapsto 1, y \mapsto 2\}}}{\emptyset \vdash C \Rightarrow \{x \mapsto 1, y \mapsto 2\}}$$

작은 보폭으로(*small-step operational semantics*)

명령문 C 와 정수식 e 의 의미는 (메모리 M 에서), 작은 보폭들

$$\begin{array}{ccc} (M, C) \rightarrow (M_1, C_1) & & (M_0, e) \rightarrow (M_1, e_1) \\ (M_1, C_1) \rightarrow (M_2, C_2) & & (M_1, e_1) \rightarrow (M_2, e_2) \\ \vdots & \text{와} & \vdots \\ (M_n, C_n) \rightarrow (M', \text{done}) & & (M_m, e_m) \rightarrow (M'', v) \end{array}$$

의 증명들.

$$\begin{array}{lcl}
M & \in & \text{Memory} = \text{Var} \xrightarrow{\text{fin}} \text{Val} \\
v & \in & \text{Val} = \mathbb{Z} \\
C & \rightarrow & \dots \mid \text{done}
\end{array}$$

$$\overline{(M, \text{skip}) \rightarrow (M, \text{done})}$$

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

$$\overline{(M, x := v) \rightarrow (M\{x \mapsto v\}, \text{done})}$$

$$\frac{(M, C_1) \rightarrow (M', C'_1)}{(M, C_1 ; C_2) \rightarrow (M', C'_1 ; C_2)}$$

$$\overline{(M, \text{done} ; C_2) \rightarrow (M, C_2)}$$

$$\frac{(M, E) \rightarrow (M, E')}{(M, \text{if } E \ C_1 \ C_2) \rightarrow (M, \text{if } E' \ C_1 \ C_2)}$$

$$\frac{}{(M, \text{if } 0 \ C_1 \ C_2) \rightarrow (M, C_1)}$$

$$\frac{}{(M, \text{if } n \ C_1 \ C_2) \rightarrow (M, C_1)} \quad n \neq 0$$

$$\frac{}{(M, \text{while } E \ C) \rightarrow (M, \text{if } E \ C ; \text{while } E \ C \ \text{skip})}$$

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

$$\frac{(M, E_1) \rightarrow (M, E'_1)}{(M, E_1 + E_2) \rightarrow (M, E'_1 + E_2)}$$

$$\frac{(M, E_2) \rightarrow (M, E'_2)}{(M, v_1 + E_2) \rightarrow (M, v_1 + E'_2)}$$

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

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

$$\overline{(\emptyset, x := 1; C_2) \rightarrow (\{x \mapsto 1\}, \text{done}; C_2)}$$

$$\overline{(\{x \mapsto 1\}, \text{done}; C_2) \rightarrow (\{x \mapsto 1\}, C_2)}$$

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

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

$$\overline{(\{x \mapsto 1\}, y := 2) \rightarrow (\{x \mapsto 1, y \mapsto 2\}, \text{done})}$$

프로그램의 의미 = 변이(transition)

변이과정 의미구조(*transition semantics*)

- ▶ 프로그램의 실행 = 무언가가 변해가는 과정
- ▶ 무한한 실행과정? 무한히 변해가기
- ▶ 예) $1+2+3$ 의 의미 = $1+2+3 \rightarrow 3+3 \rightarrow 6$

참고로,

- ▶ 이전의 작은보폭 의미 규칙들을 논리식의 증명규칙(proof rule) 대신에
- ▶ 변이규칙(transition rule)으로 볼 수 있는데, 이 때
- ▶ 한 스텝의 변화는

$$\frac{\dots}{(M, C) \rightarrow (M', C')}$$

이렇게 되면 프로그램 C 의 의미(실행과정)는

$$\frac{\dots}{(M_0, C) \rightarrow (M_1, C_1)} \Rightarrow \frac{\dots}{(M_1, C_1) \rightarrow (M_2, C_2)} \Rightarrow \dots$$

라는 전이과정으로 정의할 수 있음

문맥구조를 통해서(*evaluation-context semantics*)

변이과정 의미구조를 표현하는 한 방식

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

“실행문맥” (*evaluation context*) $K =$ 다시 쓸 부분이 정의되어 있는 프로그램

- ▶ K 가 문법적으로 정의 가능
- ▶ 다시 쓸 부분이 $[]$ (빈칸)으로 표현됨
- ▶ 편의상: 다시 쓸 부분 $[]$ 을 품은 프로그램을 “ $K[]$ ”로 표현
- ▶ 편의상: 다시 쓸 부분이 C 인 프로그램은 “ $K[C]$ ”로 표현

실행문맥 K 의 정의

$$\begin{array}{l} K \rightarrow [] \\ | x := K \\ | K ; C \\ | \text{done} ; K \\ | \text{if } K \ C \ C \\ | \text{while } K \ C \\ | K + E \\ | v + K \\ | - K \end{array}$$

다시 써야할 부분을 무엇으로 다시 써?

다시쓰기 규칙(*rewriting rule*)에 의해 정의

- ▶ 다시 쓸 곳은 다시 쓰면 되고

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

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

- ▶ 속에서 어떻게 다시 쓰여지는 가 하면:

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

$x := 1 ; y := x + 1$ 의 의미:

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

다음은,

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

다음은,

$$\frac{(\{x \mapsto 1\}, 1 + 1) \rightarrow (\{x \mapsto 1\}, 2)}{(\{x \mapsto 1\}, \text{done} ; y := [1 + 1]) \rightarrow (\{x \mapsto 1\}, \text{done} ; y := 2)}$$

다음은,

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

다음은,

$(\{x \mapsto 1, y \mapsto 2\}, \text{done} ; \text{done}) \rightarrow (\{x \mapsto 1, y \mapsto 2\}, \text{done})$.

가상의 기계를 통해서(*abstract-machine semantics*)

변이과정 의미구조의 한 방식

- ▶ 프로그램의 의미 = 프로그램이 가상의 기계에서 실행되는 과정
- ▶ 실행 과정 = 기계의 상태가 변하는 과정

정수식

$$E \rightarrow n \mid E + E \mid - E$$

가상의 기계 상태

$$\langle S, C \rangle \in Stack \times Cmd$$

여기서

$$\begin{array}{l} S \rightarrow \epsilon \quad (\text{빈 스택}) \\ \quad \mid n.S \quad (n \in \mathbb{Z}) \\ C \rightarrow \epsilon \quad (\text{빈 명령어}) \\ \quad \mid E.C \\ \quad \mid +.C \\ \quad \mid -.C \end{array}$$

기계 작동과정의 스텝은:

$$\begin{aligned}
\langle S, n.C \rangle &\rightarrow \langle n.S, C \rangle \\
\langle S, E_1 + E_2.C \rangle &\rightarrow \langle S, E_1.E_2.+C \rangle \\
\langle S, -E.C \rangle &\rightarrow \langle S, E.-C \rangle \\
\langle n_2.n_1.S, +.C \rangle &\rightarrow \langle n.S, C \rangle \quad (n = n_1 + n_2) \\
\langle n.S, -.C \rangle &\rightarrow \langle -n.S, C \rangle
\end{aligned}$$

정수식 E 의 의미는

$$\langle \epsilon, E \rangle \rightarrow \dots$$

혹은,

$$\begin{array}{l} C \rightarrow \epsilon \quad (\text{빈 명령어}) \\ | \text{push } n.C \quad (n \in \mathbb{Z}) \\ | \text{pop}.C \\ | \text{add}.C \\ | \text{rev}.C \end{array}$$

기계 작동의 한 스텝은:

$$\begin{aligned}
\langle S, \text{push } n.C \rangle &\rightarrow \langle n.S, C \rangle \\
\langle n.S, \text{pop}.C \rangle &\rightarrow \langle S, C \rangle \\
\langle n_1.n_2.S, \text{add}.C \rangle &\rightarrow \langle n.S, C \rangle \quad (n = n_1 + n_2) \\
\langle n.S, \text{rev}.C \rangle &\rightarrow \langle -n.S, C \rangle
\end{aligned}$$

이제 정수식들이 어떤 명령어들로 변환되는 지 정의:

$$\begin{aligned} \llbracket n \rrbracket &= \text{push } n \\ \llbracket E_1 + E_2 \rrbracket &= \llbracket E_1 \rrbracket . \llbracket E_2 \rrbracket . \text{add} \\ \llbracket - E \rrbracket &= \llbracket E \rrbracket . \text{rev} \end{aligned}$$

그리고 정수식 E 의 의미는

$$\langle \epsilon, \llbracket E \rrbracket \rangle \rightarrow \dots$$

- ▶ 가상의 기계를 어떻게 디자인 하는가? 그 기계의 디테일은 어느 레벨에서 정해야 하는가?
- ▶ 그에 대한 답은 프로그램의 의미를 정의하는 목적에 따라 결정됨
- ▶ 읽기: [CoCuMa87] (higher-order applicative language (e.g. ML) 실행기를 위한 기계)

예) 가상의 기계 SM3

$$\begin{array}{l} C \rightarrow \text{skip} \\ | x := E \\ | C ; C \\ | \text{if } E C C \\ E \rightarrow n \quad (n \in \mathbb{Z}) \\ | x \\ | E + E \\ | - E \end{array}$$

기계 상태

$$\langle S, M, C \rangle \in \text{Stack} \times \text{Memory} \times \text{Cmd}$$

여기서

S	$\rightarrow \epsilon$	(빈 스택)
	$n.S$	($n \in \mathbb{Z}$)
C	$\rightarrow \epsilon$	(빈 명령어)
	$E.C$	
	$+.C$	
	$-.C$	
	$\text{jmpz}(C, C).C$	
	$\text{store}(x).C$	
	$\text{load}(x).C$	

기계 작동과정의 스텝은:

$$\begin{aligned}
\langle S, M, n.C \rangle &\rightarrow \langle n.S, M, C \rangle \\
\langle S, M, E_1 + E_2.C \rangle &\rightarrow \langle S, M, E_1.E_2.+C \rangle \\
\langle v_2.v_1.S, M, +.C \rangle &\rightarrow \langle v.S, M, C \rangle \quad (v = v_1 + v_2) \\
\langle v.S, M, -.C \rangle &\rightarrow \langle -v.S, M, C \rangle \\
\langle 0.S, M, \text{jmpz}(C_1, C_2).C \rangle &\rightarrow \langle S, M, C_1.C \rangle \\
\langle v.S, M, \text{jmpz}(C_1, C_2).C \rangle &\rightarrow \langle S, M, C_2.C \rangle \quad (v \neq 0) \\
\langle v.S, M, \text{store}(x).C \rangle &\rightarrow \langle S, M\{x \mapsto v\}, C \rangle \\
\langle S, M, \text{load}(x).C \rangle &\rightarrow \langle M(x).S, M, C \rangle
\end{aligned}$$