SNU 4541.664A Program Analysis Spring 2005 Note 4

Prof. Kwangkeun Yi

계획

1. 과정을 드러내는 의미구조(*operational semantics*)
   - Style 1: 프로그램의 의미 = 증명(proof)
   - Style 2: 프로그램의 의미 = 변이(transition)

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,
  $S \cup T, S + T, S \times T, S \xrightarrow{\text{fin}} T$

$$S \xrightarrow{\text{fin}} T = \{f | f \in S' \to T, S' \overset{\text{fin}}{\subseteq} S\}$$

- 일반 함수집합 만들기 $\to$를 사용하면 곤란

Style 1: 프로그램의 의미 = 증명

structural operational semantics,
natural semantics, relational semantics

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

$$\frac{\ldots}{(M, \texttt{skip}, M')} \qquad \frac{\ldots}{(M, x \texttt{ := } E, M')}$$

$$\frac{\ldots}{(M, C_1 \texttt{ ; } C_2, M')} \qquad \frac{\ldots}{(M, \texttt{if } E \texttt{ } C_1 \texttt{ } C_2, M')}$$

$$\frac{\ldots}{(M, \texttt{while } E \texttt{ do } C, M')}$$

- 무한한 실행과정? 무한한 증명

예

명령문 C와 정수식 e의 의미는 (메모리 M 에서)

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

의 증명

- 증명 불가능 $\Leftrightarrow$ C는 메모리 M에서 무의미

큰 보폭으로(*big-step operational semantics*)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

작은 보폭으로(*small-step operational semantics*)

명령문 C와 정수식 e의 의미는 (메모리 M에서),
작은 보폭들

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

와

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

의 증명들.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

프로그램의 의미 = 변이(transition)

변이과정 의미구조(*transition semantics*)

- 프로그램의 실행 = 무언가가 변해가는 과정
- 무한한 실행과정? 무한히 변해가기
- 예) 1+2+3의 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

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

이렇게 되면 프로그램 C의 의미(실행과정)는

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

라는 전이과정으로 정의할 수 있슴

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

- K가 문법적으로 정의 가능
- 다시 쓸 부분이 $[\ ]$(빈칸)으로 표현됨
- 편의상: 다시 쓸 부분 $[\ ]$을 품은 프로그램을 "$K[\ ]$"로 표현
- 편의상: 다시 쓸 부분이 C인 프로그램은 "$K[C]$"로 표현

실행문맥 K의 정의

$$
\begin{array}{rcl}
K & \rightarrow & [\ ] \\
& | & x := K \\
& | & K\ ;\ C \\
& | & \mathsf{done}\ ;\ K \\
& | & \mathtt{if}\ K\ C\ C \\
& | & \mathtt{while}\ K\ \mathtt{do}\ C \\
& | & K + E \\
& | & v + K \\
& | & - K
\end{array}
$$

다시 써야할 부분을 무엇으로 다시 써?
다시쓰기 규칙(*rewriting rule*)에 의해 정의

- 다시 쓸 곳은 다시 쓰면 되고

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

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

- 속에서 어떻게 다시 쓰여지는 가 하면:

$$
\begin{array}{rcll}
(M, x := v) & \rightarrow & (M\{x \mapsto v\}, \mathsf{done}) & \\
(M, \mathsf{done}\ ;\ \mathsf{done}) & \rightarrow & (M, \mathsf{done}) & \\
(M, \mathtt{if}\ 0\ C_1\ C_2) & \rightarrow & (M, C_1) & \\
(M, \mathtt{if}\ v\ C_1\ C_2) & \rightarrow & (M, C_2) & (v \neq 0) \\
(M, \mathtt{while}\ 0_E\ \mathtt{do}\ C) & \rightarrow & (M, \mathsf{done}) & \\
(M, \mathtt{while}\ v_E\ \mathtt{do}\ C) & \rightarrow & (M, C\ ;\ \mathtt{while}\ E\ \mathtt{do}\ C) & (v \neq 0) \\
(M, v_1 + v_2) & \rightarrow & (M, v) & (v = v_1 + v_2) \\
(M, - v) & \rightarrow & (M, -v) & \\
(M, x) & \rightarrow & (M, M(x)) &
\end{array}
$$

$x := 1 ; y := x + 1$ 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,

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

다음은,

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

다음은,

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

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

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

기계 작동과정의 스텝은:

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

정수식 E의 의미는

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

혹은,

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

기계 작동의 한 스텝은:

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

이제 정수식들이 어떤 명령어들로 변환되는 지 정의:

$$\begin{aligned} [\![n]\!] &= \texttt{push}\, n \\ [\![E_1 \texttt{ + } E_2]\!] &= [\![E_1]\!].[\![E_2]\!].\texttt{add} \\ [\![\texttt{- } E]\!] &= [\![E]\!].\texttt{rev} \end{aligned}$$

그리고 정수식 E의 의미는

$$\langle \epsilon, [\![E]\!] \rangle \rightarrow \cdots$$

- 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

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

기계 상태

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

여기서

$$
\begin{array}{rcll}
S & \rightarrow & \epsilon & \text{(빈 스택)} \\
 & | & n.S & (n \in \mathbb{Z}) \\
C & \rightarrow & \epsilon & \text{(빈 명령어)} \\
 & | & E.C & \\
 & | & \texttt{+}.C & \\
 & | & \texttt{-}.C & \\
 & | & \texttt{jmpz}(C, C).C & \\
 & | & \texttt{store}(x).C &
\end{array}
$$

기계 작동과정의 스텝은:

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