Functions

2019

New Expressions and Statements

- Extended expressions

$$\mathscr{E} += \mathscr{X}\,\mathscr{E}^*$$

Call $f(e_1, \ldots, e_k)$

New Expressions and Statements

- Extended expressions

Call $f(e_1, \ldots, e_k)$

$$\mathscr{E} += \mathscr{X}\,\mathscr{E}^*$$

- Extended statements

$$\mathscr{S} += \texttt{\textbf{return}}\ \mathscr{E}^?$$

New Expressions and Statements

- Extended expressions

Call $f(e_1, \dots, e_k)$

$$\mathscr{E} += \mathscr{X}\,\mathscr{E}^*$$

- Extended statements

$$\mathscr{S} += \texttt{return}\;\mathscr{E}^?$$

- Extended configuration

$$\mathscr{C} = \Sigma \times \mathbb{Z}^* \times \mathbb{Z}^*$$

State $\mathscr{X} \to \mathbb{Z}$ Input Output

New Expressions and Statements

- Extended expressions

Call $f(e_1, \dots, e_k)$

$$\mathscr{E} += \mathscr{X}\,\mathscr{E}^*$$

- Extended statements

$$\mathscr{S} += \texttt{\textbf{return}}\ \mathscr{E}^?$$

- Extended configuration

Optional result

$$\mathscr{C} = \Sigma \times \mathbb{Z}^* \times \mathbb{Z}^* \times \mathbb{Z}^?$$

State $\mathscr{X} \to \mathbb{Z}$ Input Output

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad [\mathsf{Const}^{\mathscr{E}}_{bs}]$$

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad [\mathsf{Var}^{\mathscr{E}}_{bs}]$$

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad [\mathsf{Const}^{\mathscr{E}}_{bs}]$$

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad [\mathsf{Var}^{\mathscr{E}}_{bs}]$$

$$\frac{}{\Phi \vdash c \xRightarrow{A \otimes B}} \qquad [\mathsf{Binop}^{\mathscr{E}}_{bs}]$$

2

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad [\mathsf{Const}^{\mathscr{E}}_{bs}]$$

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad [\mathsf{Var}^{\mathscr{E}}_{bs}]$$

$$\frac{\Phi \vdash c \xRightarrow{A} c' = \langle \_, \_, \_, a \rangle \qquad \Phi \vdash c' \xRightarrow{B} \langle \sigma'', i'', o'', b \rangle}{\Phi \vdash c \xRightarrow{A \otimes B} \langle \sigma'', i'', o'', a \oplus b \rangle} \qquad [\mathsf{Binop}^{\mathscr{E}}_{bs}]$$

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad \left[\mathsf{Const}^{\mathscr{E}}_{bs}\right]$$

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad \left[\mathsf{Var}^{\mathscr{E}}_{bs}\right]$$

$$\frac{\Phi \vdash c \xRightarrow{A} c' = \langle \_, \_, \_, a \rangle \qquad \Phi \vdash c' \xRightarrow{B} \langle \sigma'', i'', o'', b \rangle}{\Phi \vdash c \xRightarrow{A \otimes B} \langle \sigma'', i'', o'', a \oplus b \rangle} \qquad \left[\mathsf{Binop}^{\mathscr{E}}_{bs}\right]$$

$$\frac{}{\Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})}} \qquad \left[\mathsf{Call}^{\mathscr{E}}_{bs}\right]$$

2

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, — \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad [\mathsf{Const}^{\mathscr{E}}_{bs}]$$

$$\Phi \vdash \langle \sigma, i, o, — \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad [\mathsf{Var}^{\mathscr{E}}_{bs}]$$

$$\frac{\Phi \vdash c \xRightarrow{A} c' = \langle \_, \_, \_, a \rangle \qquad \Phi \vdash c' \xRightarrow{B} \langle \sigma'', i'', o'', b \rangle}{\Phi \vdash c \xRightarrow{A \otimes B} \langle \sigma'', i'', o'', a \oplus b \rangle} \qquad [\mathsf{Binop}^{\mathscr{E}}_{bs}]$$

$$\frac{\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle}{\Phi \vdash c_0 = \left\langle \sigma_0, \_, \_, \_ \right\rangle \xRightarrow{f(\overline{e_k})}} \qquad [\mathsf{Call}^{\mathscr{E}}_{bs}]$$

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad [\mathsf{Const}^{\mathscr{E}}_{bs}]$$

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad [\mathsf{Var}^{\mathscr{E}}_{bs}]$$

$$\frac{\Phi \vdash c \xRightarrow{A} c' = \langle \_, \_, \_, a \rangle \qquad \Phi \vdash c' \xRightarrow{B} \langle \sigma'', i'', o'', b \rangle}{\Phi \vdash c \xRightarrow{A \otimes B} \langle \sigma'', i'', o'', a \oplus b \rangle} \qquad [\mathsf{Binop}^{\mathscr{E}}_{bs}]$$

$$\frac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \texttt{fun}\ f\ (\overline{a})\ \texttt{local}\ \overline{l}\ \texttt{\{}s\texttt{\}}\end{array}}{\Phi \vdash c_0 = \left\langle \sigma_0, \_, \_, \_ \right\rangle \xRightarrow{f(\overline{e_k})}} \qquad [\mathsf{Call}^{\mathscr{E}}_{bs}]$$

2

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, \text{—} \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad [\mathsf{Const}^{\mathscr{E}}_{bs}]$$

$$\Phi \vdash \langle \sigma, i, o, \text{—} \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad [\mathsf{Var}^{\mathscr{E}}_{bs}]$$

$$\frac{\Phi \vdash c \xRightarrow{A} c' = \langle \_, \_, \_, a \rangle \qquad \Phi \vdash c' \xRightarrow{B} \langle \sigma'', i'', o'', b \rangle}{\Phi \vdash c \xRightarrow{A \otimes B} \langle \sigma'', i'', o'', a \oplus b \rangle} \qquad [\mathsf{Binop}^{\mathscr{E}}_{bs}]$$

$$\frac{\begin{array}{c} \text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \mathbf{fun}\ f\ \ (\overline{a})\ \mathbf{local}\ \overline{l}\ \ \{s\} \\ \Phi \vdash \left\langle \mathbf{enter}\, \sigma_k\ (\overline{a} @ \overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, \text{—} \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle \end{array}}{\Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})} \langle \mathbf{leave}\, \sigma'\ \sigma_0, i', o', n \rangle} \qquad [\mathsf{Call}^{\mathscr{E}}_{bs}]$$

Big-Step Semantics for Expressions

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{n} \langle \sigma, i, o, n \rangle \qquad [\mathsf{Const}^{\mathscr{E}}_{bs}]$$

$$\Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{x} \langle \sigma, i, o, \sigma\, x \rangle \qquad [\mathsf{Var}^{\mathscr{E}}_{bs}]$$

$$\frac{\Phi \vdash c \xRightarrow{A} c' = \langle \_, \_, \_, a \rangle \qquad \Phi \vdash c' \xRightarrow{B} \langle \sigma'', i'', o'', b \rangle}{\Phi \vdash c \xRightarrow{A \otimes B} \langle \sigma'', i'', o'', a \oplus b \rangle} \qquad [\mathsf{Binop}^{\mathscr{E}}_{bs}]$$

$$\frac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j} c_j = \left\langle \sigma_j,\, i_j,\, o_j,\, v_j \right\rangle \\ \Phi\ f = \mathbf{fun}\ f\ \ (\overline{a})\ \mathbf{local}\ \overline{l}\ \ \{s\} \\ \color{red}{\mathtt{skip}},\Phi \vdash \left\langle \mathbf{enter}\ \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, - \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle\end{array}}{\Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})} \langle \mathbf{leave}\ \sigma'\ \sigma_0, i', o', n \rangle} \qquad [\mathsf{Call}^{\mathscr{E}}_{bs}]$$

2

New Semantics for Statements

- return

$$\frac{}{c \xRightarrow{\textbf{return } e} c'}$$

New Semantics for Statements

- return

$$\frac{[\![e]\!] \Downarrow}{c \xRightarrow{\textbf{return } e} c'}$$

New Semantics for Statements

- return

$$\frac{[\![e]\!] \Downarrow \quad \color{red}{???}}{c \xRightarrow{\textbf{return } e} c'}$$

New Semantics for Statements

- return

Suppose we fill it

$$\frac{[\![e]\!] \Downarrow \quad ???}{c \xRightarrow{\textbf{return}\ e} c'}$$

- **return** e ; S

$$\frac{}{c \xRightarrow{\textbf{return}\ e\ ;\ S} c'}$$

New Semantics for Statements

- return

$$\frac{[\![e]\!] \Downarrow \quad \textcolor{red}{???}}{c \xRightarrow{\textbf{return } e} c'}$$

Suppose we fill it

- $\texttt{return}\ e\ ;\ S$

$$\frac{c \xRightarrow{\textbf{return } e} c'' \qquad c'' \xRightarrow{S} c'}{c \xRightarrow{\textbf{return } e\ ;\ S} c'}$$

Always executes S!

I.e. **return** is not a local construction (from the control flow point of view; as **break**, **throw**, **continue**, ...)

CPS Semantics for Statements

CPS Semantics for Statements

- New component in the environment

$$K, \Phi \vdash c \xRightarrow{s} c'$$

CPS Semantics for Statements

- New component in the environment

$$K, \Phi \vdash c \xRightarrow{s} c'$$

Lack of *locality*

CPS Semantics for Statements

- New component in the environment

$$K, \Phi \vdash c \overset{s}{\Longrightarrow} c'$$

Lack of *locality*

- New meta-operator $\diamond$

$$\begin{array}{rcrcl} S & \diamond & \mathbf{skip} & = & S \\ S_1 & \diamond & S_2 & = & S_1; S_2 \end{array}$$

$$\texttt{skip}, \Phi \vdash c \xRightarrow{\texttt{skip}} c \qquad [\text{SkipSkip}]$$

CPS Rules — Basic Stmts

$$\texttt{\textbf{skip}}, \Phi \vdash c \xrightarrow{\texttt{\textbf{skip}}} c \qquad \text{[SkipSkip]}$$

$$\frac{\texttt{\textbf{skip}}, \Phi \vdash c \xrightarrow{K} c' \qquad K \neq \texttt{\textbf{skip}}}{K, \Phi \vdash c \xrightarrow{\texttt{\textbf{skip}}} c'} \qquad \text{[Skip]}$$

CPS Rules — Basic Stmts

$$\texttt{skip}, \Phi \vdash c \xRightarrow{\texttt{skip}} c \qquad [\text{SkipSkip}]$$

$$\frac{\texttt{skip}, \Phi \vdash c \xRightarrow{K} c' \qquad K \neq \texttt{skip}}{K, \Phi \vdash c \xRightarrow{\texttt{skip}} c'} \qquad [\text{Skip}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad \texttt{skip}, \Phi \vdash \langle \sigma[x \leftarrow n], i, o, - \rangle \xRightarrow{K} c'}{K, \Phi \vdash c \xRightarrow{x := e} c'} \qquad [\text{Assign}]$$

$$\texttt{skip}, \Phi \vdash c \xRightarrow{\texttt{skip}} c \qquad [\text{SkipSkip}]$$

$$\frac{\texttt{skip}, \Phi \vdash c \xRightarrow{K} c' \qquad K \neq \texttt{skip}}{K, \Phi \vdash c \xRightarrow{\texttt{skip}} c'} \qquad [\text{Skip}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad \texttt{skip}, \Phi \vdash \langle \sigma[x \leftarrow n], i, o, - \rangle \xRightarrow{K} c'}{K, \Phi \vdash c \xRightarrow{x := e} c'} \qquad [\text{Assign}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad \texttt{skip}, \Phi \vdash \langle \sigma, i, o@[n], - \rangle \xRightarrow{K} c'}{K, \Phi \vdash c \xRightarrow{\texttt{write}\ (e)} c'} \qquad [\text{Write}]$$

CPS Rules — Basic Stmts

$$\texttt{skip}, \Phi \vdash c \xRightarrow{\texttt{skip}} c \qquad [\text{SkipSkip}]$$

$$\frac{\texttt{skip}, \Phi \vdash c \xRightarrow{K} c' \qquad K \neq \texttt{skip}}{K, \Phi \vdash c \xRightarrow{\texttt{skip}} c'} \qquad [\text{Skip}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad \texttt{skip}, \Phi \vdash \langle \sigma[x \leftarrow n], i, o, - \rangle \xRightarrow{K} c'}{K, \Phi \vdash c \xRightarrow{x \texttt{:=} e} c'} \qquad [\text{Assign}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad \texttt{skip}, \Phi \vdash \langle \sigma, i, o@[n], - \rangle \xRightarrow{K} c'}{K, \Phi \vdash c \xRightarrow{\texttt{write}\ (e)} c'} \qquad [\text{Write}]$$

$$\frac{\texttt{skip}, \Phi \vdash \langle \sigma[x \leftarrow z], i, o, - \rangle \xRightarrow{K} c'}{K, \Phi \vdash \langle \sigma, z :: i, o, - \rangle \xRightarrow{\texttt{read}\ (x)} c'} \qquad [\text{Read}]$$

CPS Rules — Seq and If

$$\frac{}{K, \Phi \vdash c \xRightarrow{s_1\,;\,s_2} c'} \quad [\mathsf{Seq}]$$

CPS Rules — Seq and If

$$\frac{s_2 \diamond K,\, \Phi \vdash c \xRightarrow{s_1} c'}{K,\, \Phi \vdash c \xRightarrow{s_1;\, s_2} c'} \quad [\text{Seq}]$$

CPS Rules — Seq and If

$$\frac{s_2 \diamond K, \Phi \vdash c \xRightarrow{s_1} c'}{K, \Phi \vdash c \xRightarrow{s_1;\, s_2} c'} \quad [\mathsf{Seq}]$$

$$\frac{}{K, \Phi \vdash c \xRightarrow{\texttt{if}\ e\ \texttt{then}\ s_1\ \texttt{else}\ s_2} c'} \quad [\mathsf{IfTrue}]$$

CPS Rules — Seq and If

$$\frac{s_2 \diamond K,\, \Phi \vdash c \xRightarrow{s_1} c'}{K,\, \Phi \vdash c \xRightarrow{s_1;\, s_2} c'} \quad [\text{Seq}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \qquad K,\, \Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{s_1} c'}{K,\, \Phi \vdash c \xRightarrow{\mathbf{if}\; e\; \mathbf{then}\; s_1\; \mathbf{else}\; s_2} c'} \quad [\text{IfTrue}]$$

$$\frac{s_2 \diamond K, \Phi \vdash c \xRightarrow{s_1} c'}{K, \Phi \vdash c \xRightarrow{s_1;\, s_2} c'} \quad [\mathsf{Seq}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \qquad K, \Phi \vdash \langle \sigma, i, o, \text{—} \rangle \xRightarrow{s_1} c'}{K, \Phi \vdash c \xRightarrow{\mathbf{if}\ e\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2} c'} \quad [\mathsf{IfTrue}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n = 0 \qquad K, \Phi \vdash \langle \sigma, i, o, \text{—} \rangle \xRightarrow{s_2} c'}{K, \Phi \vdash c \xRightarrow{\mathbf{if}\ e\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2} c'} \quad [\mathsf{IfFalse}]$$

CPS Rules — While

$$\frac{}{K, \Phi \vdash c \xRightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileTrue}]$$

CPS Rules — While

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0}{K, \Phi \vdash c \xRightarrow{\texttt{while}\, e\ \texttt{do}\, s} c'} \quad [\text{WhileTrue}]$$

CPS Rules — While

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \qquad (\texttt{while}\ e\ \texttt{do}\ s) \diamond K, \Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{s} c'}{K, \Phi \vdash c \xRightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileTrue}]$$

CPS Rules — While

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \qquad (\texttt{while}\ e\ \texttt{do}\ s) \diamond K, \Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{s} c'}{K, \Phi \vdash c \xRightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileTrue}]$$

$$\frac{}{K, \Phi \vdash c \xRightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileFalse}]$$

CPS Rules — While

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \qquad (\texttt{while}\ e\ \texttt{do}\ s) \diamond K, \Phi \vdash \langle \sigma, i, o, - \rangle \xRightarrow{s} c'}{K, \Phi \vdash c \xRightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileTrue}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n = 0}{K, \Phi \vdash c \xRightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileFalse}]$$

CPS Rules — While

$$\frac{\Phi \vdash c \xrightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n \neq 0 \\ (\texttt{while}\ e\ \texttt{do}\ s) \diamond K, \Phi \vdash \langle \sigma, i, o, - \rangle \xrightarrow{s} c'}{K, \Phi \vdash c \xrightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileTrue}]$$

$$\frac{\Phi \vdash c \xrightarrow{e}_{\mathscr{E}} \langle \sigma, i, o, n \rangle \qquad n = 0 \\ \texttt{skip}, \Phi \vdash \langle \sigma, i, o, - \rangle \xrightarrow{K} c'}{K, \Phi \vdash c \xrightarrow{\texttt{while}\ e\ \texttt{do}\ s} c'} \quad [\text{WhileFalse}]$$

CPS Rules — Call and Return

$$\frac{}{K, \Phi \vdash c \xRightarrow{f(\overline{e_k})}} \quad [\text{Call}]$$

CPS Rules — Call and Return

$$\dfrac{\text{for } j \in [1..k] \ . \ \Phi \vdash c_{j-1} \xRightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle}{K, \Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})}} \quad [\text{Call}]$$

CPS Rules — Call and Return

$$\frac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xrightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \texttt{fun}\ f\ \ (\overline{a})\ \texttt{local}\ \bar{l}\ \ \texttt{\{}s\texttt{\}}\end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_\rangle \xrightarrow{f(\overline{e_k})}}\ [\text{Call}]$$

CPS Rules — Call and Return

$$\dfrac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \texttt{fun}\ f\ (\overline{a})\ \texttt{local}\ \overline{l}\ \{s\} \\ \color{red}{???}, \Phi \vdash \left\langle \textbf{enter}\ \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, \text{—} \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle\end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0,\ \_,\ \_,\ \_ \rangle \xRightarrow{f(\overline{e_k})}}\ [\text{Call}]$$

$$\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xrightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle$$

$$\Phi\ f = \texttt{fun}\ f\ \ (\overline{a})\ \texttt{local}\ \overline{l}\ \ \{s\}$$

$$???, \Phi \vdash \left\langle \textbf{enter}\ \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, - \right\rangle \xrightarrow{s} \langle \sigma', i', o', n \rangle$$

$$\frac{}{K, \Phi \vdash c_0 = \langle \sigma_0,\ \_,\ \_,\ \_ \rangle \xrightarrow{f(\overline{e_k})}} \quad [\text{Call}]$$

$$K, \Phi \vdash c \xrightarrow{\texttt{return}} \quad [\text{ReturnEmpty}]$$

8

CPS Rules — Call and Return

$$\dfrac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \texttt{fun}\ f\ \ (\overline{a})\ \texttt{local}\ \overline{l}\ \ \{s\} \\ {\color{red}???}, \Phi \vdash \left\langle \textbf{enter}\ \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, \textbf{—} \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle\end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0,\ \_,\ \_,\ \_ \rangle \xRightarrow{f(\overline{e_k})}} \quad [\text{Call}]$$

$$K, \Phi \vdash c \xRightarrow{\texttt{return}} c \quad [\text{ReturnEmpty}]$$

CPS Rules — Call and Return

$$\frac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \mathbf{fun}\ f\ \ (\overline{a})\ \mathbf{local}\ \overline{l}\ \ \{s\} \\ \color{red}{???}, \Phi \vdash \left\langle \textbf{enter}\ \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, \text{—} \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle\end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})}}\ [\text{Call}]$$

$$K, \Phi \vdash c \xRightarrow{\mathbf{return}} c \qquad [\text{ReturnEmpty}]$$

$$\frac{}{K, \Phi \vdash c \xRightarrow{\mathbf{return}\ e}}\ [\text{Return}]$$

8

CPS Rules — Call and Return

$$\frac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \texttt{fun}\ f\ \ (\overline{a})\ \texttt{local}\ \overline{l}\ \ \{s\} \\ ???, \Phi \vdash \left\langle \textbf{enter}\ \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, \textbf{—} \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle\end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})}} \quad [\text{Call}]$$

$$K, \Phi \vdash c \xRightarrow{\texttt{return}} c \qquad [\text{ReturnEmpty}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} c'}{K, \Phi \vdash c \xRightarrow{\texttt{return}\ e} c'} \quad [\text{Return}]$$

CPS Rules — Call and Return

$$\dfrac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \mathbf{fun}\ f\ \ (\overline{a})\ \mathbf{local}\ \bar{l}\ \ \{s\} \\ \mathbf{skip}, \Phi \vdash \left\langle \mathbf{enter}\ \sigma_k\ (\overline{a} @ \bar{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, - \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle\end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0,\ \_,\ \_,\ \_ \rangle \xRightarrow{f(\overline{e_k})}} \quad [\text{Call}]$$

$$K, \Phi \vdash c \xRightarrow{\mathbf{return}} c \qquad [\text{ReturnEmpty}]$$

$$\dfrac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} c'}{K, \Phi \vdash c \xRightarrow{\mathbf{return}\ e} c'} \qquad [\text{Return}]$$

CPS Rules — Call and Return

$$\frac{\begin{array}{c}\text{for } j \in [1..k] \;.\; \Phi \vdash c_{j-1} \xRightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\, f = \texttt{fun}\, f\ \ (\overline{a})\ \texttt{local}\, \overline{l}\ \ \{s\} \\ \texttt{skip}, \Phi \vdash \left\langle \textbf{enter}\, \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, \textbf{—} \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle \\ \texttt{skip}, \Phi \vdash {\color{red}???} \xRightarrow{K} c'' \end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0, \_, \_, \_ \rangle \xRightarrow{f(\overline{e_k})}} \quad [\text{Call}]$$

$$K, \Phi \vdash c \xRightarrow{\texttt{return}} c \quad [\text{ReturnEmpty}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} c'}{K, \Phi \vdash c \xRightarrow{\texttt{return}\, e} c'} \quad [\text{Return}]$$

CPS Rules — Call and Return

$$\frac{\begin{array}{c}\text{for } j \in [1..k]\ .\ \Phi \vdash c_{j-1} \xRightarrow{e_j}_{\mathscr{E}} c_j = \left\langle \sigma_j, i_j, o_j, v_j \right\rangle \\ \Phi\ f = \texttt{fun}\ f\ \ (\overline{a})\ \texttt{local}\ \overline{l}\ \ \texttt{\{}s\texttt{\}} \\ \texttt{skip}, \Phi \vdash \left\langle \textbf{enter}\ \sigma_k\ (\overline{a}@\overline{l})\ [\overline{a_j \leftarrow v_j}], i_k, o_k, \textbf{—} \right\rangle \xRightarrow{s} \langle \sigma', i', o', n \rangle \\ \texttt{skip}, \Phi \vdash \langle \textbf{leave}\ \sigma'\ \sigma_0, i', o', n \rangle \xRightarrow{K} c'' \end{array}}{K, \Phi \vdash c_0 = \langle \sigma_0,\ \_,\ \_,\ \_ \rangle \xRightarrow{f(\overline{e_k})} c''}\ [\text{Call}]$$

$$K, \Phi \vdash c \xRightarrow{\texttt{return}} c \qquad [\text{ReturnEmpty}]$$

$$\frac{\Phi \vdash c \xRightarrow{e}_{\mathscr{E}} c'}{K, \Phi \vdash c \xRightarrow{\texttt{return}\ e} c'}\ [\text{Return}]$$

Functions X86-32

X86-32

- Standard caller code:

- Standard caller code:

$$\begin{array}{ll} \text{push} & arg_n \\ \text{push} & arg_{n-1} \\ \ldots & \\ \text{push} & arg_1 \\ \text{call} & < callee\ name > \\ \text{addl} & n*4,\ \%esp \end{array}$$

X86-32

- Standard caller code:

$$
\begin{array}{ll}
\text{push} & arg_n \\
\text{push} & arg_{n-1} \\
\ldots & \\
\text{push} & arg_1 \\
\text{call} & < callee\ name > \\
\text{addl} & n*4,\ \%esp
\end{array}
$$

- Result $\rightarrow$ $\%eax$

- Standard caller code:

push	arg_n
push	arg_{n-1}
$\ldots$	
push	arg_1
call	$< callee\ name >$
addl	$n * 4,\ \%esp$

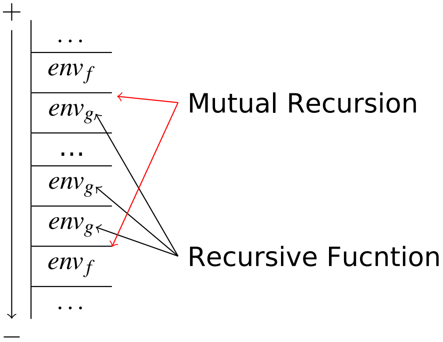
$\ldots$
env_f
env_g
...
env_g
env_g
env_f
$\ldots$

- Result $\rightarrow$ $\%eax$

- Standard caller code:

$$\begin{array}{ll}
\text{push} & arg_n \\
\text{push} & arg_{n-1} \\
\ldots & \\
\text{push} & arg_1 \\
\text{call} & < callee\ name > \\
\text{addl} & n*4,\ \%esp
\end{array}$$

- Result $\rightarrow$ $\%eax$



Frames

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack

Frames

Parameters

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack

Frames

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack

Parameters

Return Value(-s)

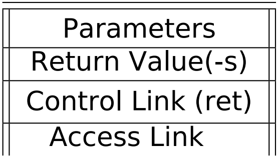
Frames

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack

Parameters
Return Value(-s)
Control Link (ret)

Frames

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack



Frames

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack

Parameters
Return Value(-s)
Control Link (ret)
Access Link
Saved Machine State

Frames

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack

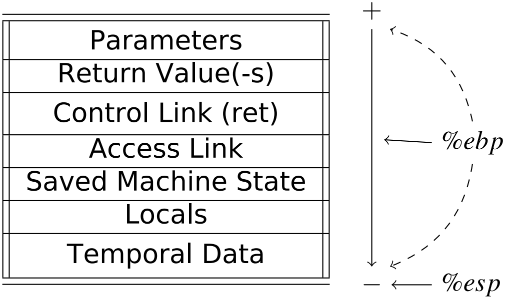
Parameters
Return Value(-s)
Control Link (ret)
Access Link
Saved Machine State
Locals

Frames

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack

Parameters
Return Value(-s)
Control Link (ret)
Access Link
Saved Machine State
Locals
Temporal Data

Each activation has an ***activation record*** (*frame* or *memory display*) on the call stack



Application Binary Interface

- ABI
- EABI
- Calling convention

Standard prologue
X86-32:

pushl $\%ebp \leftarrow$ Save Callers ebp
movl $\%esp, \%ebp \leftarrow$ Set up our ebp
subl S$, \%esp \leftarrow$ Set up our esp

Locals Size

Epilogue

Standard Epilogue X86-32:

movl $\%ebp, \%esp$
popl $\%ebp$
ret

Epilogue

Standard Epilogue X86-32:

```
movl    %ebp, %esp
popl    %ebp
ret
```

Registers

- EAX, EDX, ECX — caller-saved registers
- EBX, EDI, ESI (, and EBP) — callee-saved registers
- EIP, ESP (, and EBP) — special purpose registers

Compiling SM to x86-32

- Call in SM: $\textbf{call}_{SM}$ f
  How many arguments we have to copy from symbolic stack?

  $addl$ S. $\%esp$ How to compute S?

- Call in SM: **call**$_{SM}$ f
  How many arguments we have to copy from symbolic stack?

  $addl$ S. $\%esp$

  How to compute S?

  Solutions:

  - Lookup for corresponding **BEGIN** and get the info

    it is very fragile

Compiling SM to x86-32

- Call in SM: **call**$_{SM}$ f
  How many arguments we have to copy from symbolic stack?

  $addl$ S. $\%esp$ How to compute S?

  Solutions:

  - Lookup for corresponding **BEGIN** and get the info

    it is very fragile

  - **call**$_{SM}$ f $\longrightarrow$ **call**$_{SM}$ (f, n)

    useless for SM,
    but is very useful for compilation

- Call in SM: **call**$_{SM}$ f
  How many arguments we have to copy from symbolic stack?

  $addl$ S. $\%esp$ How to compute S?

  Solutions:
  - Lookup for corresponding **BEGIN** and get the info
    it is very fragile
  - **call**$_{SM}$ f $\longrightarrow$ **call**$_{SM}$ (f, n)
    useless for SM,
    but is very useful for compilation
- For SM there is no difference between function and procedure calls
  In x86-32 there is: do we need to move the result from symbolyc stack to $\%eax$?

- Call in SM: $\mathbf{call}_{SM}$ f
  How many arguments we have to copy from symbolic stack?

  $addl$ S. $\%esp$ How to compute S?

  Solutions:

  - Lookup for corresponding **BEGIN** and get the info

    it is very fragile
  - $\mathbf{call}_{SM}$ f $\longrightarrow$ $\mathbf{call}_{SM}$ (f, n)

    useless for SM,
    but is very useful for compilation

- For SM there is no difference between function and procedure calls
  In x86-32 there is: do we need to move the result from symbolyc stack to $\%eax$? flag: procedure/function

  $\mathbf{call}_{SM}$ (f, n) $\longrightarrow$ $\mathbf{call}_{SM}$ (f, n, p)

Compiling SM to x86-32

- In SM we generate **END** for each **return**; In x86 we can generate epilogue once

- **BEGIN**$_{SM}$ has to be accomplished with function name in order to find the locals size
  But how to calculate this constant during code generation?

Compiling SM to x86-32

- In SM we generate **END** for each **return**; In x86 we can generate epilogue once

- We need to return the result: flag: true iff return in *Expr*

$$\mathbf{RET}_{SM} \longrightarrow \mathbf{RET}_{SM}\ (f)$$

- $\mathbf{BEGIN}_{SM}$ has to be accomplished with function name in order to find the locals size
  But how to calculate this constant during code generation?

- In SM we generate **END** for each **return**; In x86 we can generate epilogue once

- We need to return the result:

flag: true iff return in *Expr*

$$\textbf{RET}_{SM} \longrightarrow \textbf{RET}_{SM}\ (f)$$

- **BEGIN**$_{SM}$ has to be accomplished with function name in order to find the locals size
  But how to calculate this constant during code generation?
  — Use GASM symbolic constants: $.set\ \ < name > < constant >$

- In SM we generate **END** for each **return**; In x86 we can generate epilogue once
- We need to return the result: flag: true iff return in *Expr*

$$\textbf{RET}_{SM} \longrightarrow \textbf{RET}_{SM}\ (f)$$

- **BEGIN**$_{SM}$ has to be accomplished with function name in order to find the locals size
  But how to calculate this constant during code generation?
  — Use GASM symbolic constants: $.set\ \ <name><constant>$
- See **class env** for details

Generating code for $\mathbf{CALL}_{SM}$

$\mathbf{CALL}_{SM}$ **(f, n, p)**

Generating code for **CALL**$_{SM}$

CALL$_{SM}$ **(f, n, p)**

1. Move **n** values on X86 stack

care of the order!

Generating code for **CALL**$_{SM}$

CALL$_{SM}$ **(f, n, p)**

1. Move **n** values on X86 stack

   care of the order!

2. **call** f

Generating code for **CALL**$_{SM}$

CALL$_{SM}$ **(f, n, p)**

1. Move **n** values on X86 stack

care of the order!

2. **call** f

3. $addl\ 4 * \mathbf{n},\ \%esp$

Generating code for **CALL**$_{SM}$

CALL$_{SM}$ **(f, n, p)**

1. Move **n** values on X86 stack

   care of the order!

2. **call** f
3. $addl\ 4 * \mathbf{n},\ \%esp$
4. if **!p** then push the result on symbolic stack from $\%eax$

Generating code for **CALL**$_{SM}$

CALL$_{SM}$ **(f, n, p)**

1. Move **n** values on X86 stack

   care of the order!

2. **call** f

3. $addl\ 4*$**n**$,\ \%esp$

4. if **!p** then push the result on symbolic stack from $\%eax$

5. Save registers on x86 stack before the call and restore after

   look into **env#live_registers**