Imperative Languages I

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, A-4040 Linz, Austria

Wolfgang.Schreiner@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/people/schreine

Imperative Languages

Essential Features:

- Sequential execution,
- Implicit data structure (the “store”)
  - Existence indpendent of any program,
  - Not mentioned in language syntax,
  - Phrases may access it and update it.

Relationship between store and programs:

1. Critical for evaluation of phrases.

   Phrase meaning depends on store.

2. Communication between phrases.

   Phrases deposit values in store for use by other phrases; sequencing mechanism establishes communication order.

3. Inherently “large” argument.

   Only one copy existing during execution.

A Language with Assignment

- Declaration-free Pascal subset.
- Program = sequence of *commands*
- $\mathbf{C}$: Command $\rightarrow$ *Store*$_\perp$ $\rightarrow$ *Store*$_\perp$
  - A command produces a new store from its store argument.
  - Command might not terminate ("loop")

    $\rightarrow$ $\mathbf{C}[[C]]s = \perp$.
  - Followup commands will not evaluate

    $\rightarrow$ $\mathbf{C}[[C]]$:*Store*$_\perp$ $\rightarrow$ *Store*$_\perp$ *is strict.*
  - Command sequencing is store composition

    $\rightarrow$ $\mathbf{C}[[C_1; C_2]](s) = \mathbf{C}[[C_2]](\mathbf{C}[[C_1]]s)$
    $\rightarrow$ $\mathbf{C}[[C_1; C_2]] = \mathbf{C}[[C_1]] \circ \mathbf{C}[[C_2]]$

(See Schmidt, Figures 5.1 and 5.2)

Valuation Functions

- **P**: Program $\rightarrow$ *Nat* $\rightarrow$ *Nat*$_\perp$

  Program maps input number to an answer number; non-termination is possible (codomain includes $\perp$).

- **C**: Command $\rightarrow$ *Store*$_\perp$ $\rightarrow$ $Store_\perp$

  Command maps store into a new store; predecessor command may not have terminated (domain includes $\perp$) and command may not terminate (codomain includes $\perp$).

- **E**: Expression $\rightarrow$ *Store* $\rightarrow$ *Nat*

  Expression maps store into natural number.

- **B**: Bool-exp $\rightarrow$ *Store* $\rightarrow$ *Tr*

  Boolean expression maps store into truth value.

- **N**: Numeral $\rightarrow$ *Nat*

  Numeral yields a natural value.

Program Denotation

- How to understand a program?
- A possibility is to compute its denotation with a particular input argument.

  P: Program $\rightarrow$ *Nat* $\rightarrow$ $Nat_\perp$
- Various results for various inputs.

  Natural number or $\perp$ (non-termination)

```
Z:=1;
if A=0 then diverge;
Z:=3.
```

$\Downarrow$ **P** (*two*)

$Denotation$

Simplification

$\mathbf{P}[[\text{Z:=1; } \textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3.}]](\textit{two})$
$= \text{let } s_1 = (\textit{update } [[\text{A}]] \textit{ two newstore})$
$\quad s' = \mathbf{C}[[\text{Z:=1; } \textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3}]]s_1$
$\quad \text{in } \textit{access } [[\text{Z}]] \ s'$
$= \text{let } s_1 = (\ [\ [[\text{A}]] \mapsto \textit{two}\ ]\ \textit{newstore})$
$\quad s' = \underline{\mathbf{C}[[\text{Z:=1; } \textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3}]]s_1}$
$\quad \text{in } \textit{access } [[\text{Z}]] \ s'$

$\mathbf{C}[[\text{Z:=1; } \textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3}]]s_1$
$= (\underline{\lambda}\text{s.}\mathbf{C}[[\textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3}]]$
$\quad (\mathbf{C}[[\text{Z:=1}]]s))s_1$
$= \mathbf{C}[[\textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3}]](\underline{\mathbf{C}[[\text{Z:=1}]]s_1})$

$\mathbf{C}[[\text{Z:=1}]]s_1$
$= (\underline{\lambda}\text{s.}\textit{update } [[\text{Z}]] \ (\mathbf{E}[[1]]s) \ s)s_1$
$= \textit{update } [[\text{Z}]] \ (\mathbf{E}[[1]]s_1) \ s_1$
$= \textit{update } [[\text{Z}]] \ (\mathbf{N}[[1]]) \ s_1$
$= \textit{update } [[\text{Z}]] \ \textit{one} \ s_1$
$= [\ [[\text{Z}]] \mapsto \textit{one}\ ]s_1$
$= s_2$

Simplification

$\mathbf{C}[\![\textbf{if } \mathsf{A}{=}0 \textbf{ then diverge}; \mathsf{Z}{:=}3]\!](\underline{\mathbf{C}[\![\mathsf{Z}{:=}1]\!]s_1})$
$= \mathbf{C}[\![\textbf{if } \mathsf{A}{=}0 \textbf{ then diverge}; \mathsf{Z}{:=}3]\!]s_2$
$= (\underline{\lambda} s.\mathbf{C}[\![\mathsf{Z}{:=}3]\!](\mathbf{C}[\![\textbf{if } \mathsf{A}{=}0 \textbf{ then diverge}]\!]s)s_2$
$= (\underline{\lambda} s.\mathbf{C}[\![\mathsf{Z}{:=}3]\!]((\underline{\lambda} s.\mathbf{B}[\![\mathsf{A}{=}0]\!]s \rightarrow$
$\quad \mathbf{C}[\![\textbf{diverge}]\!]s \; [] \; s) \; s))s_2$
$= \mathbf{C}[\![\mathsf{Z}{:=}3]\!]((\underline{\lambda} s.\mathbf{B}[\![\mathsf{A}{=}0]\!]s \rightarrow$
$\quad \mathbf{C}[\![\textbf{diverge}]\!]s \; [] \; s)s_2)$
$= \mathbf{C}[\![\mathsf{Z}{:=}3]\!](\underline{\mathbf{B}[\![\mathsf{A}{=}0]\!]s_2} \rightarrow \mathbf{C}[\![\textbf{diverge}]\!]s_2 \; [] \; s_2)$

$\mathbf{B}[\![\mathsf{A}{=}0]\!]s_2$
$= (\lambda \; s.\mathbf{E}[\![\mathsf{A}]\!]s \; \textit{equals} \; \mathbf{E}[\![0]\!]s)s_2$
$= \mathbf{E}[\![\mathsf{A}]\!]s_2 \; \textit{equals} \; \mathbf{E}[\![0]\!]s_2$
$= \underline{(\textit{access} \; [\![\mathsf{A}]\!] \; s_2)} \; \textit{equals zero}$

$\textit{access} \; [\![\mathsf{A}]\!] \; s_2$
$= s_2 \; [\![\mathsf{A}]\!]$
$= (\; [\; [\![\mathsf{Z}]\!] \mapsto \textit{one} \;][\; [\![\mathsf{A}]\!] \mapsto \textit{two} \;] \; \textit{newstore} \;) \; [\![\mathsf{A}]\!]$
$= ([\; [\![\mathsf{A}]\!] \mapsto \textit{two} \;] \; \textit{newstore}) \; [\![\mathsf{A}]\!]$
$= \textit{two}$

Simplification

$\underline{(\mathit{access}\ [[\mathsf{A}]]\ s_2)}\ \mathit{equals\ zero}$
$= \mathit{two\ equals\ zero}$
$= \mathit{false}$

$\mathbf{C}[[\mathsf{Z{:=}3}]](\underline{\mathbf{B}[[\mathsf{A{=}0}]]s_2} \rightarrow \mathbf{C}[[\mathbf{diverge}]]s_2\ []\ s_2)$
$= \mathbf{C}[[\mathsf{Z{:=}3}]](\mathit{false} \rightarrow \mathbf{C}[[\mathbf{diverge}]]s_2\ []\ s_2)$
$= \mathbf{C}[[\mathsf{Z{:=}3}]]s_2$
$= (\underline{\lambda} s.\mathit{update}\ [[\mathsf{Z}]]\ (\mathbf{E}[[3]]s)\ s)s_2$
$= \mathit{update}\ [[\mathsf{Z}]]\ (\mathbf{E}[[3]]s_2)\ s_2$
$= \mathit{update}\ [[\mathsf{Z}]]\ (\mathbf{N}[[3]])\ s_2$
$= \mathit{update}\ [[\mathsf{Z}]]\ \mathit{three}\ s_2$
$= [\ [[\mathsf{Z}]] \mapsto \mathit{three}\ ]s_2$

Simplification

let s_1 = ([[[A]] $\mapsto$ *two*] *newstore*)
s_2 = [[[Z]] $\mapsto$ *one*]s_1
s' = **C**[[Z:=1; **if** A=0 **then diverge**; Z:=3]]s_1
in *access* [[Z]] s'
= let s_1 = ([[[A]] $\mapsto$ *two*] *newstore*)
s_2 = [[[Z]] $\mapsto$ *one*]s_1
s' = [[[Z]] $\mapsto$ *three*]s_2
in *access* [[Z]] s'

access [[Z]] s'
= *access* [[Z]] [[[Z]] $\mapsto$ *three*]s_2
= [[[Z]] $\mapsto$ *three*]s_2 [[Z]]
= *three*

Program Denotation

Program plus input $\longrightarrow$ result.

```
Z:=1;
if A=0 then diverge;
Z:=3.
```

$\Downarrow$ **P** (*two*)

let s_1 = ([[[A]] $\mapsto$ *two*] *newstore*)
s_2 = [[[Z]] $\mapsto$ *one*]s_1
s_3 = ((*access* [[A]] s_2) *equals zero*) $\rightarrow \perp$ [] s_2
s' = [[[Z]] $\mapsto$ *three*]s_3
in *access* [[Z]] s'

$\parallel$

three

Intermediate form (only partial simplification) delivers more insight!

Simplification II

$\textbf{P}[[\text{Z:=1; } \textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3.}]](\textit{zero})$
$= \text{let } s_3 = [\ [[\text{A}]] \mapsto \textit{zero}\ ]\textit{newstore}$
$\quad s' = \textbf{C}[[\text{Z:=1; } \textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3}]]s_3$
$\quad \text{in } \textit{access}\ [[\text{Z}]]\ s'$
$= \text{let } s_3 = [\ [[\text{A}]] \mapsto \textit{zero}\ ]\textit{newstore}$
$\quad s_4 = [\ [[\text{Z}]] \mapsto \textit{one}\ ]s_3$
$\quad s' = \textbf{C}[[\textbf{if} \text{ A=0 } \textbf{then diverge}\text{; Z:=3}]]s_4$
$\quad \text{in } \textit{access}\ [[\text{Z}]]\ s'$
$= \text{let } s_3 = [\ [[\text{A}]] \mapsto \textit{zero}\ ]\textit{newstore}$
$\quad s_4 = [\ [[\text{Z}]] \mapsto \textit{one}\ ]s_3$
$\quad s_5 = \underline{\textbf{C}[[\textbf{if} \text{ A=0 } \textbf{then diverge}]]s_4}$
$\quad s' = \textbf{C}[[\text{Z:=3}]]s_5$
$\quad \text{in } \textit{access}\ [[\text{Z}]]\ s'$

$\textbf{C}[[\textbf{if} \text{ A=0 } \textbf{then diverge}]]s_4$
$= \textbf{B}[[\text{A=0}]]s_4 \rightarrow \textbf{C}[[\textbf{diverge}]]s_4\ []\ s_4$
$= \textit{true} \rightarrow \textbf{C}[[\textbf{diverge}]]s_4\ []\ s_4$
$= \textbf{C}[[\textbf{diverge}]]s_4$
$= (\underline{\lambda} s.\bot)s_4$
$= \bot$

Simplification II

let s_3 = [[[A]] $\mapsto$ *zero*]*newstore*

s_4 = [[[Z]] $\mapsto$ *one*]s_3

s_5 = **C**[[**if** A=0 **then diverge**]]s_4

s' = **C**[[Z:=3]]s_5

in *access* [[Z]] s'

= let s_3 = [[[A]] $\mapsto$ *zero*]*newstore*

s_4 = [[[Z]] $\mapsto$ *one*]s_3

$s_5 = \perp$

s' = **C**[[Z:=3]]s_5

in *access* [[Z]] s'

= let s' = **C**[[Z:=3]]$\perp$

in *access* [[Z]] s'

= let s' = (λs.*update* [[Z]] (**E**[[3]]s))$\perp$

in*access* [[Z]] s'

= let $s' = \perp$

in*access* [[Z]] s'

= *access* [[Z]] $\perp$

$= \perp$

Program Equivalence

$$\mathbf{C}[[\text{X:=0; Y:=X+1}]] \stackrel{?}{=} \mathbf{C}[[\text{Y:=1;X:=0}]]$$

- $\mathbf{C}$: $\mathit{Store}_\perp \rightarrow \mathit{Store}_\perp$
- Show $\mathbf{C}[[\text{C}_1]]s = \mathbf{C}[[\text{C}_2]]s$ for every s.
- $\mathbf{C}[[\text{C}_1]]\perp = \perp = \mathbf{C}[[\text{C}_2]]\perp$.

Assume proper store s.

$\mathbf{C}[[\text{X:=0; Y:=X+1}]]s$
$= \mathbf{C}[[\text{Y:=X+1}]](\mathbf{C}[[\text{X:=0}]]s)$
$= \mathbf{C}[[\text{Y:=X+1}]]([\ [[\text{X}]] \mapsto \mathit{zero}\ ]s)$
$= \mathit{update}\ [[\text{Y}]]\ (\mathbf{E}[[\text{X+1}]]([\ [[\text{X}]] \mapsto \mathit{zero}\ ]s))$
$\quad [\ [[\text{X}]] \mapsto \mathit{zero}\ ]s$
$= \mathit{update}\ [[\text{Y}]]\ \mathit{one}\ [\ [[\text{X}]] \mapsto \mathit{zero}\ ]s$
$= [\ [[\text{Y}]] \mapsto \mathit{one}\ ][\ [[\text{X}]] \mapsto \mathit{zero}\ ]s$
$= s_1$

Program Equivalence

$\mathbf{C}[[\text{Y:=1; X:=0}]]s$
$= \mathbf{C}[[\text{X:=0}]](\mathbf{C}[[\text{Y:=1}]]s)$
$= \mathbf{C}[[\text{X:=0}]]([\ [[\text{Y}]] \mapsto \textit{one}\ ]s)$
$= [\ [[\text{X}]] \mapsto \textit{zero}\ ][\ [[\text{Y}]] \mapsto \textit{one}\ ]s = s_2$

- $s_1 \stackrel{?}{=} s_2$.
- s_1, s_2: *Id* $\longrightarrow$ *Nat*
- Show $s_1(\textit{id}) = s_2(\textit{id})$ for every *id*.

1. <u>*id* = [[X]]</u>: $s_1[[\text{X}]] = ([\ [[\text{Y}]] \mapsto \textit{one}\ ][\ [[\text{X}]] \mapsto \textit{zero}\ ]s)[[\text{X}]]$ $= ([[\text{X}]] \mapsto \textit{zero}\ ]s)[[\text{X}]] = \textit{zero} = ([\ [[\text{X}]] \mapsto \textit{zero}\ ][\ [[\text{Y}]] \mapsto \textit{one}\ ]s)[[\text{X}]] = s_2[[\text{X}]]$.
2. <u>*id* = [[Y]]</u>: $s_1[[\text{Y}]] = ([\ [[\text{Y}]] \mapsto \textit{one}\ ][\ [[\text{X}]] \mapsto \textit{zero}\ ]s)[[\text{Y}]]$ $= \textit{one} = ([\ [[\text{Y}]] \mapsto \textit{one}\ ]s)[[\text{Y}]] = ([\ [[\text{X}]] \mapsto \textit{zero}\ ][\ [[\text{Y}]] \mapsto \textit{one}\ ]s)[[\text{Y}]] = s_2[[\text{Y}]]$.
3. <u>*id* = [[I]]</u>: $s_1[[\text{I}]] = ([\ [[\text{Y}]] \mapsto \textit{one}\ ][\ [[\text{X}]] \mapsto \textit{zero}\ ]s)[[\text{I}]]$ $= ([\ [[\text{X}]] \mapsto \textit{zero}\ ]s)[[\text{I}]] = s[[\text{I}]] = ([\ [[\text{Y}]] \mapsto \textit{one}\ ]s)[[\text{I}]]$ $= ([\ [[\text{X}]] \mapsto \textit{zero}\ ][\ [[\text{Y}]] \mapsto \textit{one}\ ]s)[[\text{I}]] = s_2[[\text{I}]]$.

Programs Are Functions

- Simplifications were operational-like.
- Answer computed from program and input.

Z:=1;
if A=0 **then diverge**;
Z:=3.

$\Downarrow$ **P**

$\lambda n.$let s_1 = ([[[A]] $\mapsto n$] *newstore*)
s_2 = [[[Z]] $\mapsto$ *one*]s_1
s_3 = ((*access* [[A]] s_2) *equals zero*) $\rightarrow \perp$ [] s_2
s' = [[[Z]] $\mapsto$ *three*]s_3
in *access* [[Z]] s'

Study denotation without sample input!

Programs are Functions

$$\|$$

$$\begin{array}{l}\lambda n.\text{let } s_1 = (\ [\ [[\text{A}]] \mapsto n]\ \textit{newstore}) \\ \quad s_2 = [\ [[\text{Z}]] \mapsto \textit{one}\ ]s_1 \\ \quad s_3 = (n\ \textit{equals zero}) \rightarrow \bot\ []\ s_2 \\ \quad s' = [\ [[\text{Z}]] \mapsto \textit{three}\ ]s_3 \\ \text{in } \textit{access}\ [[\text{Z}]]\ s'\end{array}$$

$$\|$$

$$\begin{array}{l}\lambda n.\text{let } s_1 = (\ [\ [[\text{A}]] \mapsto n]\ \textit{newstore}) \\ \quad s_2 = [\ [[\text{Z}]] \mapsto \textit{one}\ ]s_1 \\ \quad s_3 = (n\ \textit{equals zero}) \rightarrow \bot\ []\ s_2 \\ \quad s' = (n\ \textit{equals zero}) \rightarrow \bot\ []\ [\ [[\text{Z}]] \mapsto \textit{three}\ ]s_3 \\ \text{in } \textit{access}\ [[\text{Z}]]\ s'\end{array}$$

$$\|$$

$$\begin{array}{l}\lambda n.\text{let } s_1 = (\ [\ [[\text{A}]] \mapsto n]\ \textit{newstore}) \\ \quad s_2 = [\ [[\text{Z}]] \mapsto \textit{one}\ ]s_1 \\ \text{in } (n\ \textit{equals zero}) \rightarrow \bot \\ \quad []\ \textit{access}\ [[\text{Z}]]\ [\ [[\text{Z}]] \mapsto \textit{three}\ ]s_2\end{array}$$

$$\|$$

$$\lambda n.(n\ \textit{equals zero}) \rightarrow \bot\ []\ \textit{three}$$

Program Denotation

Z:=1;
if A=0 **then diverge**;
Z:=3.

$\Downarrow$ **P**

$\lambda n.(n$ *equals zero*$) \rightarrow \bot$ [] *three*

- Denotation extracts *essence* of a program.
- Store disappears

  Temporary data structure; not contained in the input/output relation of a program.

- Transformation resembles compilation.
- Simplification resembles optimization.