# COMS22202: 2015/16

# Language Engineering

**Dr Oliver Ray**
**(csxor@Bristol.ac.uk)**

**Department of Computer Science**
**University of Bristol**

**Thursday 25th February, 2016**

# Reminder of Expected Time Allocation

½ Unit: 10 CP
≈100 hrs

Class: 3*12
=36 hrs

Homework: 2*12
=24 hrs

CWK2
≈16 hrs

Revision: 2*12
≈24 hrs

Note: the hardest part of course is probably over the next few weeks

Note: tomorrow I will set CWK2 PT1 (due at the end of next week)

# Program Statements: p 85

**Syntax:** $S \; ::= \; x := a \mid \mathbf{skip} \mid S_1 \, ; S_2 \mid \mathbf{if} \; b \; \mathbf{then} \; S_1 \; \mathbf{else} \; S_2 \mid \mathbf{while} \; b \; \mathbf{do} \; S$

**Semantics:** $\mathcal{S}_{\mathrm{ds}} \; : \; \mathbf{Stm} \rightarrow ( \mathbf{State} \hookrightarrow \mathbf{State} )$

- **The denotational (or direct) semantics of statements are functions, called *state transformers*, mapping states (before) to states (after)**
- **In general, state transformers are *partial* functions as programs (with loops) may not terminate from some states: e.g. while true do skip**
- **Partial functions can be viewed as mapping some of their inputs to an undefined output denoted** undef **(or $\perp$ in Haskell books)**
- **Note: the semantics of arithmetics and Booleans would be partial if we had division (even integer division) as division by 0 is undefined**
- **We use FIX to denote the least fixpoint of a function between state transformers wrt. the partial order of subset inclusion over the set of states upon which a state transformer is defined**
- **A function between state transformers is also called a *functional***

# Semantic Definitions: p 86

$\mathcal{S}_{ds}[[\ x := a\ ]]\ s \quad = \quad s\ [\ x \mapsto \mathcal{A}\ [[\ a\ ]]\ s\ ]$

$\mathcal{S}_{ds}[[\ \mathbf{skip}\ ]] \quad = \quad \mathbf{id}$

$\mathcal{S}_{ds}[[\ S_1\ ;\ S_2\ ]] \quad = \quad \mathcal{S}_{ds}[[\ S_2\ ]] \circ \mathcal{S}_{ds}[[\ S_1\ ]]$

$\mathcal{S}_{ds}[[\ \mathbf{if}\ \ b\ \mathbf{then}\ \ S_1\ \ \mathbf{else}\ \ S_2\ ]] \quad = \quad \mathbf{cond}\ (\mathcal{B}\ [[\ b\ ]],\ \mathcal{S}_{ds}[[\ S_1\ ]],\ \mathcal{S}_{ds}[[\ S_2\ ]]\ )$

$\mathcal{S}_{ds}[[\ \mathbf{while}\ b\ \mathbf{do}\ S\ ]] \quad = \quad \mathbf{FIX}\ F\ \text{where}\ F\ g = \mathbf{cond}\ (\mathcal{B}\ [[\ b\ ]],\ g \circ \mathcal{S}_{ds}[[\ S\ ]],\ \mathbf{id}\ )$

# Assignment: p 86

$$\mathcal{S}_{\text{ds}}[\![\, x := a \,]\!]\; s \;=\; s\,[\, x \mapsto \mathcal{A}[\![\, a \,]\!]\, s \,]$$

$$\mathcal{S}_{\text{ds}}[\![\, x := a \,]\!] \;=\; \lambda s \,.\; s\,[\, x \mapsto \mathcal{A}[\![\, a \,]\!]\, s \,]$$

$$\text{If } \mathcal{S}_{\text{ds}}[\![\, x := a \,]\!]\; s\; v \;=\; \begin{cases} \mathcal{A}[\![\, a \,]\!]\, s & \text{if } v = x \\ s\; v & \text{otherwise} \end{cases}$$

# skip: p 86

$\mathcal{S}_{ds}[\![\, \textbf{skip} \,]\!] \;=\; \textbf{id}$

$\mathcal{S}_{ds}[\![\, \textbf{skip} \,]\!] \;=\; \lambda\, \textbf{s} \,.\, \textbf{s}$

$\mathcal{S}_{ds}[\![\, \textbf{skip} \,]\!]\; \textbf{s} \;=\; \textbf{s}$

# Sequences: p 86

$$\mathcal{S}_{\text{ds}}[\![ \, S_1 \, ; \, S_2 \, ]\!] \;=\; \mathcal{S}_{\text{ds}}[\![ \, S_2 \, ]\!] \circ \mathcal{S}_{\text{ds}}[\![ \, S_1 \, ]\!]$$

$$\begin{aligned}
\mathcal{S}_{\text{ds}}[\![ \, S_1 \, ; \, S_2 \, ]\!] \, s \;&=\; (\mathcal{S}_{\text{ds}}[\![ \, S_2 \, ]\!] \circ \mathcal{S}_{\text{ds}}[\![ \, S_1 \, ]\!]) \, s \\
&=\; \mathcal{S}_{\text{ds}}[\![ \, S_2 \, ]\!] \, ( \mathcal{S}_{\text{ds}}[\![ \, S_1 \, ]\!] \, s ) \\
&=\; s'' \;\; \textbf{where} \;\; s'' = \mathcal{S}_{\text{ds}}[\![ \, S_2 \, ]\!] \, s' \; \textbf{and} \; s' = \mathcal{S}_{\text{ds}}[\![ \, S_1 \, ]\!] \, s
\end{aligned}$$

**n.b.** $\mathcal{S}_{\text{ds}}[\![ \, S_1 \, ; \, S_2 \, ]\!] \, s \;=\; \underline{\text{undef}}$ **if** $s' = \underline{\text{undef}}$ **or** $s'' = \underline{\text{undef}}$

# Conditionals: p 87

$$\mathcal{S}_{ds}[\![\ \mathbf{if}\ \ b\ \mathbf{then}\ \ S_1\ \ \mathbf{else}\ \ S_2\ ]\!]\ \ =\ \ \mathbf{cond}\ (\mathcal{B}\ [\![\ b\ ]\!],\ \mathcal{S}_{ds}[\![\ S_1\ ]\!],\ \mathcal{S}_{ds}[\![\ S_2\ ]\!]\ )$$

$$\mathcal{S}_{ds}[\![\ \mathbf{if}\ \ b\ \mathbf{then}\ \ S_1\ \ \mathbf{else}\ \ S_2\ ]\!]\ s\ \ =\ \ \mathbf{cond}\ (\mathcal{B}\ [\![\ b\ ]\!],\ \mathcal{S}_{ds}[\![\ S_1\ ]\!],\ \mathcal{S}_{ds}[\![\ S_2\ ]\!]\ )\ s$$

$$= \begin{cases} \mathcal{S}_{ds}[\![S_1]\!]\ s & \text{if}\ \ \mathcal{B}[\![b]\!]\ s\ = tt \\ \mathcal{S}_{ds}[\![S_2]\!]\ s & \text{otherwise} \end{cases}$$

n.b. $\mathcal{S}_{ds}[\![\ \mathbf{if}\ \ b\ \mathbf{then}\ \ S_1\ \ \mathbf{else}\ \ S_2\ ]\!]\ s\ = \underline{\text{undef}}$

if $\mathcal{B}[\![b]\!]\ s\ = tt$ and $\mathcal{S}_{ds}[\![S_1]\!]\ s = \underline{\text{undef}}$

or $\mathcal{B}[\![b]\!]\ s\ = ff$ and $\mathcal{S}_{ds}[\![S_2]\!]\ s = \underline{\text{undef}}$

# Loops: p 87-8

$\mathcal{S}_{ds}$[[ while b do S ]] = FIX F where F g = cond ($\mathcal{B}$ [[ b ]], g ° $\mathcal{S}_{ds}$[[ S ]], id)

$\mathcal{S}_{ds}$[[ while b do S ]] = $\mathcal{S}_{ds}$[[ if b then ( S ; while b do S ) else skip ]]

= cond ( $\mathcal{B}$[[b]], $\mathcal{S}_{ds}$[[S ; while b do S]], $\mathcal{S}_{ds}$[[skip]] )

= cond ( $\mathcal{B}$[[b]], $\mathcal{S}_{ds}$[[while b do S]] ° $\mathcal{S}_{ds}$[[S]], id)

$\in$ fix ( $\lambda$ g . cond ( $\mathcal{B}$[[b]], g ° $\mathcal{S}_{ds}$[[S]], id ) )

= FIX ( $\lambda$ g . cond ( $\mathcal{B}$[[b]], g ° $\mathcal{S}_{ds}$[[S]], id ) )

***as shown in the book using a lot of math which you won't be expected to understand (p89-111)***

**n.b.** $\mathcal{S}_{ds}$[[ while b do S ]] s = <u>undef</u> if (FIX F) s = <u>undef</u>

# The Functional of a Loop: cf. p 89-111

- The functional $F = \lambda\, g\,.\, \mathrm{cond}\ (\mathcal{B}[[b]],\ g \circ \mathcal{S}_{ds}[[S]],\ \mathrm{id})$ is referred to as ***the functional of the loop*** **while b do S**
- This functional can be seen as a means of finding better and better approximations to the semantics of the loop as it can be shown

  $F^n(\varnothing)$ is a correct semantics for all states from which the loop ends in *fewer than* n iterations (and is undefined otherwise)

- It can be shown FIX gives the correct semantics for all possible n

  $$\mathrm{FIX}\ F\ =\ \bigcup_{n\geq 0} F^n(\varnothing)\ =\ \varnothing \cup F(\varnothing) \cup F(F(\varnothing)) \cup F(F(F(\varnothing))) \cup \ldots$$

- A *direct characterisation* of F is any equivalent mathematical expression that does not contain any semantic functions
- Note the functions mentioned above have the following types:

  $\mathrm{FIX} : (\mathrm{State} \hookrightarrow \mathrm{State}) \to (\mathrm{State} \hookrightarrow \mathrm{State}) \to (\mathrm{State} \hookrightarrow \mathrm{State})$

  $F\ \ : (\mathrm{State} \hookrightarrow \mathrm{State}) \to (\mathrm{State} \hookrightarrow \mathrm{State})$

  $g\ \ : \mathrm{State} \hookrightarrow \mathrm{State}$

# (Semantic) Equivalence: p112

Two program statements $S_1$ and $S_2$ are (denotationally) equivalent iff

$$\mathcal{S}_{\mathbf{ds}}[[\ S_1\ ]] = \mathcal{S}_{\mathbf{ds}}[[\ S_2\ ]]$$