Lab Session 6
Static Typing

Interpretation and Compilation of Languages

Nova School of Science and Technology
Mário Pereira `mjp.pereira@fct.unl.pt`

Version of April 14, 2025

Preamble – The While Language

Before we begin the exercises, let us recap the definition of the WHILE language as presented during lectures.

Syntax. Figure 1 gives the syntax of the WHILE language.

e	::=		EXPRESSIONS
	|	v	integer or Boolean constant
	|	x	variable
	|	$e \oplus e$	binary operator ($+$, $<$, ...)
	|	(e, e)	pair construction
	|	`fst` e	first projection
	|	`snd` e	second projection
s	::=		STATEMENTS
	|	x`:=`e	assignment
	|	`if (`e`) s else s`	conditional
	|	`while (`e`) s`	loop
	|	`s; s`	sequence
	|	`skip`	do nothing
	|	`repeat s until` e	repeat loop
v	::=		VALUES
	|	n	integer value
	|	b	Boolean value
	|	(v, v)	pair value

Figure 1: Syntax of the While language.

Operational Semantics. As in the lectures, we define the big steps operational semantics for our WHILE language, using environments (functions from variables to values) which we denote by σ. For instance, $\sigma = \{a \mapsto 34, b \mapsto 55\}$ is an environment that maps 34 to the variable a and 55 to the variable b. We denote by $\sigma\{x \mapsto v\}$ the environment in which the value for x has been set to v. We then state the big step operational semantics of WHILE by defining two relations $\sigma, e \Downarrow v$ and $\sigma, s \Downarrow \sigma$ that read respectively "in environment σ, expression e has value v" and "in environment σ, the evaluation of statement s terminates and leads to environment σ'". Figure 2 defines the operational semantics of the WHILE language, via inference rules (as explained in the course).

$$\frac{}{\sigma, n \Downarrow n} \qquad \frac{}{\sigma, b \Downarrow b} \qquad \frac{x \in \mathtt{dom}(\sigma)}{\sigma, x \Downarrow \sigma(x)}$$

$$\frac{\sigma, e_1 \Downarrow n_1 \quad \sigma, e_2 \Downarrow n_2 \quad n \stackrel{\text{def}}{=} n_1 \oplus n_2 \quad \oplus \in \{+, -, *, \ldots\}}{\sigma, e_1 \oplus e_2 \Downarrow n}$$

$$\frac{\sigma, e_1 \Downarrow b_1 \quad \sigma, e_2 \Downarrow b_2 \quad b \stackrel{\text{def}}{=} b_1 \oplus b_2 \quad \oplus \in \{<, \ldots\}}{\sigma, e_1 \oplus e_2 \Downarrow b}$$

$$\frac{\sigma, e_1 \Downarrow v_1 \quad \sigma, e_2 \Downarrow v_2}{\sigma, (e_1, e_2) \Downarrow (v_1, v_2)} \qquad \frac{\sigma, e \Downarrow (v_1, v_2)}{\sigma, \mathtt{fst}\ e \Downarrow v_1} \qquad \frac{\sigma, e \Downarrow (v_1, v_2)}{\sigma, \mathtt{snd}\ e \Downarrow v_2}$$

Semantics for expressions

$$\frac{}{\sigma, \mathtt{skip} \Downarrow \sigma} \qquad \frac{\sigma, s_1 \Downarrow \sigma_1 \quad \sigma_1, s_2 \Downarrow \sigma_2}{\sigma, s_1 ; s_2 \Downarrow \sigma_2}$$

$$\frac{\sigma, e \Downarrow v}{\sigma, x := e \Downarrow \sigma\{x \mapsto v\}}$$

$$\frac{\sigma, e \Downarrow \mathtt{true} \quad \sigma, s_1 \Downarrow \sigma_1}{\sigma, \mathtt{if}\ (e)\ s_1\ \mathtt{else}\ s_2 \Downarrow \sigma_1} \qquad \frac{\sigma, e \Downarrow \mathtt{false} \quad \sigma, s_2 \Downarrow \sigma_2}{\sigma, \mathtt{if}\ (e)\ s_1\ \mathtt{else}\ s_2 \Downarrow \sigma_2}$$

$$\frac{\sigma, e \Downarrow \mathtt{true} \quad \sigma, s \Downarrow \sigma_1 \quad \sigma_1, \mathtt{while}\ (e)\ s \Downarrow \sigma_2}{\sigma, \mathtt{while}\ (e)\ s \Downarrow \sigma_2} \qquad \frac{\sigma, e \Downarrow \mathtt{false}}{\sigma, \mathtt{while}\ (e)\ s \Downarrow \sigma}$$

$$\frac{\sigma, s \Downarrow \sigma'' \qquad \sigma', e \Downarrow \mathtt{false} \qquad \sigma'', \mathtt{repeat}\ s\ \mathtt{until}\ e \Downarrow \sigma'}{\sigma, \mathtt{repeat}\ s\ \mathtt{until}\ e \Downarrow \sigma'}$$

$$\frac{\sigma, s \Downarrow \sigma' \qquad \sigma', e \Downarrow \mathtt{true}}{\sigma, \mathtt{repeat}\ s\ \mathtt{until}\ e \Downarrow \sigma'}$$

Semantics for statements

Figure 2: Big-step operational semantics of the While language.

1 Typing

We now consider typing of the language WHILE from the previous part. To this end we extend types with pair type $\tau \times \tau$:

τ	::=		**type**
		$\mathtt{int}$	type of integer values
	$\mid$	$\mathtt{bool}$	type of Boolean values
	$\mid$	$\mathtt{ref}\ \tau$	type of variables
	$\mid$	$(\tau \times \tau)$	pair type

The Figure 3 below gives the typing rules for expressions and statements of WHILE, except for the rules for pairs and new loop constructions that are missing:

$$\frac{}{\Gamma \vdash n : \mathtt{int}} \qquad \frac{}{\Gamma \vdash b : \mathtt{bool}} \qquad \frac{x \in \mathrm{dom}(\Gamma)}{\Gamma \vdash x : \Gamma(x)}$$

$$\frac{\Gamma \vdash e_1 : \mathtt{int} \quad \Gamma \vdash e_2 : \mathtt{int}}{\Gamma \vdash e_1 + e_2 : \mathtt{int}} \qquad \frac{\Gamma \vdash e_1 : \mathtt{int} \quad \Gamma \vdash e_2 : \mathtt{int}}{\Gamma \vdash e_1 < e_2 : \mathtt{bool}} \qquad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 = e_2 : \mathtt{bool}}$$

Typing for the While expressions (except for the pair constructions).

$$\frac{}{\Gamma \vdash \mathtt{skip}} \qquad \frac{\Gamma \vdash s_1 \quad \Gamma \vdash s_2}{\Gamma \vdash s_1; s_2} \qquad \frac{\Gamma \vdash x : \mathtt{ref}\ \tau \quad \Gamma \vdash e : \tau}{\Gamma \vdash x := e}$$

$$\frac{\Gamma \vdash e : \mathtt{bool} \qquad \Gamma \vdash s_1 \qquad \Gamma \vdash s_2}{\Gamma \vdash \mathtt{if}\ (e)\ s_1\ \mathtt{else}\ s_2} \qquad \frac{\Gamma \vdash e : \mathtt{bool} \qquad \Gamma \vdash s}{\Gamma \vdash \mathtt{while}\ (e)\ s}$$

Typing for the While statements (except for new loop constructions).

Figure 3: Static typing for the WHILE language.

Exercise 1. Complete the typing for expressions by defining the typing rules for each of the pair (e_1, e_2), $\mathtt{fst}\ e$ and $\mathtt{snd}\ e$ expressions.

□

Exercise 2. Consider the program

$$\mathtt{if}\ (\mathtt{fst}\ x > 0)\ x := (\mathtt{snd}\ x, \mathtt{fst}\ x + \mathtt{snd}\ x)\ \mathtt{else}\ x := 0$$

Explain why this program is rejected by the static type system of WHILE.

□

Exercise 3. Define the typing rule for the repeat loop $\mathtt{repeat}\ s\ \mathtt{until}\ e$.

□