# Lab Session 2 Operational Semantics and Interpreters

Interpretation and Compilation of Languages

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

Version of March 14, 2025

# 1 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.

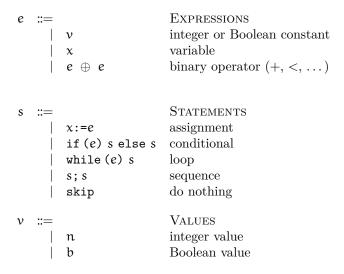


Figure 1: Syntax of the While language.

**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  $\sigma$ . For instance,  $\sigma = \{\alpha \mapsto 34, b \mapsto 55\}$  is an environment that maps 34 to the variable  $\alpha$  and 55 to the variable  $\alpha$ . We denote by  $\sigma\{x \mapsto \nu\}$  the environment in which the value for  $\alpha$  has been set to  $\alpha$ .

We then state the big step operational semantics of While by defining two relations  $\sigma, e \downarrow \nu$  and  $\sigma, s \downarrow \sigma$  that read respectively "in environment  $\sigma$ , expression e has value  $\nu$ " and "in environment  $\sigma$ , the evaluation of statement s terminates and leads to environment  $\sigma'$ ". Figure 2 defines the operational semantics of the While language, via inference rules (as explained in the course).

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

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

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

$$\text{Semantics for expressions}$$

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

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

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

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

$$\frac{\sigma, e \Downarrow \text{false} \quad \sigma, s_2 \Downarrow \sigma_2}{\sigma, \text{if (e) } s_1 \text{ else } s_2 \Downarrow \sigma_2}$$

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

Semantics for statements

 $\frac{\sigma, e \Downarrow \mathtt{false}}{\sigma, \mathtt{while}\; (e) \; s \Downarrow \sigma}$ 

### 1.1 Semantically Equivalent Programs

 $\sigma, e \Downarrow \texttt{true} \quad \sigma, s \Downarrow \sigma_1 \quad \sigma_1, \texttt{while} \; (e) \; s \Downarrow \sigma_2$ 

 $\sigma$ , while (e) s  $\psi$   $\sigma_2$ 

Two statements  $s_1$  and  $s_2$  of the While language are said to be semantically equivalent if for all states  $\sigma$  and  $\sigma'$ 

$$\sigma, s_1 \Downarrow \sigma'$$
 if and only if  $\sigma, s_2 \Downarrow \sigma'$ 

Exercise 1. Show that the statement

if (e) 
$$s_1$$
 else  $s_2$ 

is semantically equivalent to

if 
$$(\neg e)$$
  $s_2$  else  $s_1$ 

where  $\neg e$  stands for the Boolean negation of the value of expression e. You may consider the following operational semantic rules for  $\sigma, \neg e \downarrow b$ :

$$\frac{\sigma, e \Downarrow \mathtt{true}}{\sigma, e \Downarrow \mathtt{false}} \qquad \frac{\sigma, e \Downarrow \mathtt{false}}{\sigma, e \Downarrow \mathtt{true}}$$

#### Exercise 2. Show that the statement

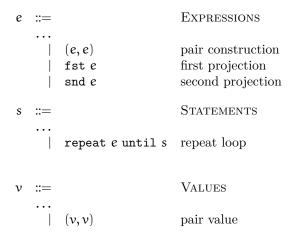
is semantically equivalent to

# 1.2 Extensions to the While Language

Consider we extend the While language with

- pair constructors (e, e) and get the left and the right pair components denoted respectively by fst e and snd e;
- repeat s until b loops.

The syntax of the language is extended as follows:



#### 1.3 Pairs

**Exercise 3.** Complete the definition of  $\sigma, e \downarrow \nu$  by giving the inference rules for the operation semantics of pairs. Concretely, define the inference rule for evaluating (e1, e2) and the inference rules to evaluate fst e and snd e in a given environment  $\sigma$ .

Exercise 4. Give a derivation for the evaluation

$$\{x\mapsto (21,34)\}, \texttt{if (fst } x>0) \ x := (\texttt{snd } x,\texttt{fst } x+\texttt{snd } x) \ \texttt{else } x := 0 \ \Downarrow \ \{x\mapsto (34,55)\}$$

Exercise 5. Consider the following While program:

```
i := 0;
p := 0;
while (i < 5)
p := (i + 1, p);
i := i + 1</pre>
```

Consider  $\sigma$  to be the final environment to which the above program evaluates. What is the value of  $\sigma(p)$ ?

- A (((((0,1),2),3),4),5)
- B (5, (4, (3, (2, (1, 0)))))
- C(0,(1,(2,(3,(4,5)))))
- D there is not such value, since the program produces a runtime error.

# 1.4 Repeat Loop

Exercise 6. We now consider the repeat s until e loop. Intuitively, its semantics is that we execute the statement s then check whether the condition e holds: if e evaluates to true, we leave the loop. Otherwise, e evaluates to false, and the loop is executed again. Define the inferences rules for each case of the repeat s until e.

**Exercise 7.** As one may notice, repeat-loops do not extend the power of the language. In fact, they can be simply rewritten (for instance by a compiler phase) using sequence and a while loop. Explain how this rewriting can be done. (You can simply write an equation repeat s until e = ...)