Homework 5

The purpose of this assignment is to make yourself familiar with operational semantics and substitutions. This step is important for preparing you for the next stage of our JavaScript interpreter implementation.

For submission instructions and the due date, please see the `README.md` file.

Problem 1 Operational Semantics (26 Points)

In this problem, we will practice how to use the inference rules of the operational semantics to evaluate expressions. Moreover, we will practice how to define inference rules for new language constructs.

(a) Show the derivation tree for the evaluation of the following expressions according to the big-step operational semantics given in Figure 4.1 of the course notes. Start your evaluation with the environment $env = \{x \mapsto 3, y \mapsto -2\}$. **(5 Points)**

   (i) $3 * x + 2$

   (ii) $\textbf{const}\ b = 2 + y; b\ \texttt{?}\ x\ \texttt{:}\ y$

   Annotate each inference rule application with the name of the rule that is being used. See also Section 4.1.3 of the course notes for an example of such a derivation tree.

(b) Show the evaluation steps of the following expressions according to the small-step operational semantics given in figures 4.2 and 4.3 of the course notes. Start your evaluation with the empty environment $env = \emptyset$ (i.e., the environment that does not bind any variables).

   (i) $3 + (1\ \texttt{\&\&}\ 5)$

   (ii) $\textbf{const}\ x = 2 + 1; x * 0\ \texttt{?}\ x\ \texttt{:}\ x + x$

   For each evaluation step, name the top-level inference rule that has been applied. If the top-level rule is a search rule, then underline the subexpression where the do rule has been applied in that step and provide the name of that do rule. You can number the arrows for the individual evaluation steps so that you can provide the information about the used rules separately from the evaluation sequence. **(5 Points)**

   Example: Consider the expression $\textbf{const}\ y = 2 * 2; y + 3$. The evaluation steps for this expression are as follows:

$$\textbf{const}\ y = \underline{2 * 2}; y + 3 \xrightarrow{a} \textbf{const}\ y = 4; \underline{y} + 3 \xrightarrow{b} \textbf{const}\ y = 4; \underline{4 + 3} \xrightarrow{c} \textbf{const}\ y = 4; 7 \xrightarrow{d} 7$$

   Top-level rule and do rule used in each step:

   - a: SearchConstDecl1, DoTimes
   - b: SearchConstDecl2, DoVar

- c: SearchConstDecl2, DoPlus
- d: DoConstDecl

(c) Consider the small-step semantics of $e_1 + e_2$ given in figures 4.2 and 4.3 of the course notes (i.e., the rules DoPlus, SearchBop1 and SearchBop2). These rules realize a left-to-right evaluation order of such expressions (i.e., first e_1 is evaluated and then e_2). Provide alternative rules for the semantics of $e_1 + e_2$ that realize a right-to-left evaluation order of such expressions (i.e., first e_2 is evaluated and then e_1). **Hint:** You only need to change the search rules. **(5 Points)**

(d) The JavaScript sequencing operator `,` allows us to compose expressions sequentially. The intended semantics of an expression e_1 `,` e_2 is that we first evaluate e_1 and then we evaluate e_2. The entire expression then evaluates to the result of e_2. That is, the result of e_1 is discarded. We only evaluate e_1 to observe the side effects of its evaluation (e.g., printing). For example, the expression $(2 + 3)$ `,` $(3 + 4)$, should evaluate to 7, which is the result of the subexpression after `,`. The first subexpression $(2 + 3)$ should be evaluated, but its result is discarded.

Provide inference rules for both the big-step and small-step SOS of e_1 `,` e_2 that formalize the semantics of described above. Provide both the do and search rules for the small-step SOS of the new operator. Your search rules should enforce the correct evaluation order for the subexpressions e_1 and e_2.

Here are the templates for the rules that you need to complete. Big-step rule:

$$\frac{?}{\mathit{env} \vdash e_1 \texttt{ , } e_2 \Downarrow ?}\ \textsc{EvalSeq}$$

Small-step rules:

$$\frac{?}{\mathit{env} \vdash e_1 \texttt{ , } e_2 \rightarrow ?}\ \textsc{SearchSeq} \qquad \frac{?}{\mathit{env} \vdash v_1 \texttt{ , } e_2 \rightarrow ?}\ \textsc{DoSeq}$$

(5 Points)

(e) For each of the following JavaScript programs, provide the result of evaluating the program according to the dynamic binding semantics given in figures 5.1 and 5.2 of the course notes. You do not need to show the intermediate results of the evaluation. However, for each application of the EvalVar rule during evaluation, describe (1) the using occurrence of the variable to which the rule is applied to, (2) the value that is retrieved for that variable from the current environment in the rule application, and (3) where in the program the respective variable was bound to this value during evaluation. **(6 Points)**

(i) Program:

```
const x = 2;
const g = function(y) (x + y);
const f = function(y) (g(y));
f(3)
```

(ii) Program:

```
const x = 2;
const g = function(x) (function(y)(x + y));
const f = function(y) (g(y)(y));
f(3)
```

Example: Consider the following program

```
1 const x = 2;
2 const g = function(y) (x + y);
3 const f = function(x) (g(x));
4 f(3)
```

This program evaluates to the value `6`. During evaluation, the EvalVar rule is applied five times as follows:

- the first application is for the using occurrence of `f` on line 4. In this case, `f` is bound to the value `function(x) (g(x))` in the **`const`** declaration on line 3.
- the second application is for the using occurrence of `g` on line 3. In this case, `g` was bound to the value `function(y) (x + y)` in the **`const`** declaration on line 2.
- the third application is for the using occurrence of `x` in the definition of function `f` on line 3. In this case, `x` was bound to the value `3` in the call to `f` on line 4.
- the forth application is for the using occurrence of `x` in the definition of function `g` on line 2. Again, `x` was bound to the value `3` in the call to `f` on line 4.
- the fifth application is for the using occurrence of `y` in `g` on line 2. This occurrence of `y` was bound to the value `3` in the call to `g` on line 3.

Problem 2 Substitutions (14 Points)

In the following, let $x, y, z \in \mathit{Var}$ be distinct variables. Given the expressions

$$
\begin{aligned}
e_1 &= (x * y) + 4 \\
e_2 &= \textbf{const}\ y = y\,\texttt{;}\, x + y \\
e_3 &= \textbf{const}\ x = (\textbf{function}\,\texttt{(}y\texttt{)}\,(x(y)))\,\texttt{;}\, x(y)
\end{aligned}
$$

compute the following independent substitutions:

(a) $e_1[3/x]$

(b) $e_1[3/z]$

(c) $e_2[3/x]$

(d) $e_2[3/y]$

(e) $e_3[(y(2))/y]$

(f) $e_3[(y(x))/x]$

You only need to compute the substitutions. Expression evaluation is not required. **Be careful:** you may need to rename bound variables to avoid variable capturing for some of the substitutions.

Example: $e = \textbf{const}\ x = (\textbf{function}\,(y)\ x + y)\,\textbf{;}\ x(y)$
$e[(y+2)/x] = \textbf{const}\ x = (\textbf{function}\,(z)\ (y+2) + z)\,\textbf{;}\ x(y)$