Lab 3 Writeup

Justin Cai

February 24, 2018

2a. Consider the JAVASCRIPTY program:

```
const x = 1
const f = (y) => x+y;
const g = (x) => f(1);
g(2);
```

Under dynamic scoping, `g(2)` will evaluate to 3, while under static scoping, it will evaluate to 2. In static scoping, call `f` will allways return $1 + y$, where y is the function parameter. Therefore, `g(x)` calls `f(1)`, which will return 2. However, with dynamic scoping, calling `g(x)` will call `f(1)` under the environment that maps x to whatever parameter passed into the function. In this case, we will change x to be 2 so then `f(y)` will return $2+y$, and will return 3.

3d. For $e \to e'$ to be deterministic it must the case that if $e \to e'$ and $e \to e''$, then $e' = e''$. This simply means there is only one way to take one single step of evaluation. For example,

$$\frac{e_1 \to e_1'}{e_1 + e_2 \to e_1' + e_2} \qquad \frac{e_2 \to e_2'}{e_1 + e_2 \to e_1 + e_2'}$$

These rules for addition would not be deterministic, as there are two possible ways to step addition.

4. The evaluation order of $e_1 + e_2$ is to evaluate e_1 down to a value and then e_2. To get the opposite order, the rules would be

$$\frac{e_2 \to e_2'}{e_1 + e_2 \to e_1 + e_2'} \qquad \frac{e_1 \to e_1'}{e_1 + v_2 \to e_1' + v2}$$

These rule will fully evaluate e_2 down to a value before evaluating e_1.

5. a. In languages like C++, when dereferencing a pointer, you should usually check to make sure the pointer is not null before doing so. Short circuiting lets you write expressions like `p != NULL && *p == x`. This will never throw an error as if p is null, it won't be dereferenced in the second part.

   b. Yes, as in DOANDTRUE, if the lefthand side is true, the expression will step to v_1, and e_2 won't be evaluated.