Lab 3 Writeup

Rasmi Lamichhane
Alex Ring

<u>**Question 2a**</u>

Give one test case that behaves differently under dynamic scoping versus static scoping (and does not crash). Explain the test case and how they behave differently in your write-up.

```
const b = 5;

const foo = function (x){
   const a = b + 5;
   return a;
}

const boo = function (x){
   const b = 2;
   return foo(1);
}

console.log(foo(1))
console.log(boo(1))
```

The above function will print 10 and 10 for static scoping and 10 and 7 for dynamic scoping. This is because with static scoping the first declaration of b = 5 will be used in the foo function, so when you enter the foo function from the call in boo, it will use the b=5 binding for b (because it is in the next highest scope) and return 10. Where as with dynamic scoping the b=2 binding in the boo function will be the newest thing on the stack and be used for b when foo is called. In the static instance we return 5 + 5, and in the dynamic case we return 5 + 2.

<u>**Question 3d**</u>

Explain whether the evaluation order is deterministic as specified by the judgment form *e* -> *e*'.

Yes the evaluation order is deterministic as specified by the judgement form e -> e'. Each of the step judgement forms only has a singular next step, and according to the course notes: "If our rules have this property then we say that our reduction system is deterministic. In other words, there is always at most one 'next' step."

Question 4

Consider the small-step operational semantics for JAVASCRIPTY shown in Figures 7, 8, and 9. What is the evaluation order for e1 + e2? Explain. How do we change the rules obtain the opposite evaluation order?

The evaluation order for e1 + e2 is a value check and step procedure until we hit two values. First you check e1 to be a value, if it is not you recursively step on e1 until you arrive at a value. If e1 is a value, you check e2 to be a value. If e2 is not a value you recursively step on e2 until you arrive at a value. Once both e1 and e2 are values you do the addition. If we wanted to reverse this order we would simply change the judgement forms to step on e2 before e1.

Question 5

In this question, we will discuss some issues with short-circuit evaluation.

(a) Concept. Give an example that illustrates the usefulness of short-circuit evaluation. Explain your example.

The example of short-circuiting would be the function "and". For example for and function traditionally we can say if a==b return true and if a!=b then return false. However, we can skip most of the step and just say if true return B(a==b) else false.

(b) JAVASCRIPTY. Consider the small-step operational semantics for JAVASCRIPTY shown in Figures 7, 8, and 9. Does e1 && e2 short circuit? Explain.

Yes e1 && e2 short circuit. In the judgement form you are checking a value v1 to be true or false. If the value if true you then return e2 for further evaluation, but if it is false you simply stop evaluating. Avoiding unneeded work of evaluating the expression e2.