Lab 3 Write-Up

Yuxiang Wang

Principles of Programming Languages

CSCI 3155-Fall 2017

2(a): 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.

Answer:

We give a test case like:

```
const x = 8
const g = function() { return x+x }
const h = function(x) { return g() }
console.log(h(5))
```

We got the results: Static scoping returns: 16.
Dynamic scoping returns: 10.

The two results are total differences because x defined as 8 outside the function and the static scoping take the initial environment, and take x=8 to the function and ignores the console.log(h(5)) .The dynamic scoping take the new environment , console.log(h(5)) is able to change the values of original x, so x=5 is use for function h and h need use function g, ignores const x = 8.

3(b). Explain whether the evaluation order is deterministic as specified by the judgment form e -> e'.

Answer: Yes, the evaluation order is deterministic as specified by the judgment form e -> e'. Because e -> e' is evaluate from left to right, so the evaluation order is always evaluate left at first and then evaluate right. We know the eval function is use for interpret expression, so the function will always evaluate in the same order and result in the same output if the same expression runs multiple times.

4. Evaluation Order. 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?

Answer: The evaluation order for e1+e2 is evaluate e1 at first until e1 is a value, then evaluate e2 until e2 is a value, finally do the +. We need check the e1 first, then e2, make sure they are string or number , then do the operator.
To change the evaluation order, we need add e2->e2' to rule like

$$\frac{e2 \to e2'}{e1 + e2 \to e1 + e2'}$$

,because we need evaluate the small step at the first, so when we add e2->e2', we do the e2 at the first, so that mean we do the right at first, so opposite evaluation order.

5. Short-Circuit Evaluation. 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.

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

Answer:

(a)
```
if( d!=0 && a/d)
{
        Print success
};
```

We know d can’t be 0 of a/d, If we don’t have d!=0, the a/d should be crash when d=0. Then we add d!=0, If d equals to zero, then it would short-circuit and skip the "a/d" so that we would not receive an error for dividing by zero.

(b) e1 && e2 short circuits because the program checks the e1 at first, if e1 is false, just returned, needn’t check e2. If e1 is valid, then get the value of e1 then evaluate e2.