CSCI 3155, Recitation 10

1 Type

We will summarize some of the features of the following different evaluation strategies that we have in IMPNAMESMALLA: call-by-value (both val and var annotations), call-by-reference (ref) and call-by-name (name). We will focus on the effects these evaluation strategies have on memory, side effects and at what step exactly memory operations occur.

We will do so by looking at the following simple example:

```plaintext
var x = 5 in
  ((pann y: t) => y = y+1; y)(print(7); x);
print(x)
```

This program captures the following 4 cases:

- call-by-value(val)
  ```plaintext
  var x = 5 in
  ((y: t) => y+1; y)(print(7); x);
  print(x)
  ```

- call-by-value(var)
  ```plaintext
  var x = 5 in
  ((var y: t) => y = y+1; y)(print(7); x);
  print(x)
  ```

- call-by-reference(ref)
  ```plaintext
  var x = 5 in
  ((ref y: t) => y+1; y)(print(7); x);
  print(x)
  ```
• call-by-name(name)

    var x = 5 in
    ((name y: t) => y+1; y)(print(7); x);
    print(x)

Before we begin, could you summarize the result of evaluating the function calls? What are the side effects (changes to memory and printed expressions)? At what steps do actual memory operations occur (dereferences)?

Note that the parser we have would convert the program above into the following form:

    ((var x: Int) =>
     ((pann y: t) => y = y+1; y)(print(7); x);
     print(x))(5)

Given the evaluation rule DoApplyFunVar, we get:

    \langle M = \{a_x \rightarrow 5\}, ((pann y: t) => y = y+1; y)(print(7); *a_x); print(*a_x)\rangle

What do you think the result of evaluating the following program is:

    \langle M = \{a_x \rightarrow 5\}, ((name y: t) => y = y+1; y)(print(7); *a_x); print(*a_x)\rangle?

What would happen with a call-by-name evaluation strategy if we never use the parameter in the function body expression? Is this efficient?

Summary:

<table>
<thead>
<tr>
<th></th>
<th>Call-by-value</th>
<th>Call-by-reference</th>
<th>Call-by-name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>val</strong></td>
<td>Eval arg to value (side effects)</td>
<td>Eval arg to value (side effects)</td>
<td>Lazy eval</td>
</tr>
<tr>
<td><strong>var</strong></td>
<td>Eval arg to value (side effects)</td>
<td>Eval arg to Deref expression (side effects)</td>
<td>“Reuse” addr for param</td>
</tr>
<tr>
<td><strong>ref</strong></td>
<td>No memory mutation</td>
<td>“Reuse” addr for param</td>
<td>No memory mutation</td>
</tr>
<tr>
<td></td>
<td>New addr for fun param</td>
<td>Memory changes are persistent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New addr not accessible after call</td>
<td></td>
<td>Eval arg everytime</td>
</tr>
</tbody>
</table>

Page 2