Assignment 5

Andrei Ungur - 260690364, Jastaj Virdee - 260689027

November 30, 2017

1 Q 1.1

Free variables are defined as variables which don't immediately evaluate to a value. In other words they are variables that are not yet bound. For example, in let x = 1 in x + 3, "x" is free in the expression "x + 3" but bound by x = 1.

In a pair of two expressions, the set of free variables will be the union between the variables of each expression.
For instance, for pair = (e1,e2), FV(pair) = FV(e1) U FV(e2).

In a let with a "match" instead of a "value", i.e.: (x,y) = e instead of simply x = e, the logic will be similar to the original let. However, since we're dealing with a pair instead of a single value, we need to remove all free occurences of both x and y in e2.
Formally,
FV(Let(e1,x),e2) = FV(e1) U (FV(e2)/x)
FV(Let(Match(e1, x, y),e2) = FV(e1) U $FV(e2/x, y)$

2 Q 1.3

The substitution of x in a pair is straight forward. It is similar to the simple substitution of x in an expression e, except now we have two expressions e1 and e2. Therefore, the output will be a new pair in which we've substituted x in both of the expression of the input pair.

In Let with a match (Let y1,y2 = e1 in e2), we first replace x in the first expression e1. If y1 or y2 are equal to x, then it will be sufficient to replace them in e1 (what they evaluate to in e1 will also determine their value in e2).

Otherwise, we would have to replace y1,y2 in the free variables of e2 we replace y1 and y2 with. If y1 is a member of the free variables of e', we need to ensure that y2 is also a free variable within e', and vice-versa. Otherwise, if only y1 is a free variable within e', then we have a similar case to the regular Let.

$[e'/x]$Pair(e1,e2) $\rightarrow$ Pair($[e'/x]$e1,$[e'/x]e2$)

$[e'/x]$(let y1,y2 = $e1$ in $e2$) $\rightarrow$ let y1,y2 = $[e'/x]e1$ in $[e'/x]e2$

3 Q 1.5

Pattern matching will use the type Match(e, x, y) where we have x and y, two variables, and e, the expression to which the pattern matching evaluates to.

To use pattern matching with Let, we would need: Let(Match(e1,x,y),e2). This is the equivalent to: Let x,y = (pattern matching on y with evaluates to e1) in e2.

Well-typed expression:
We know that e2 must evaluate to the same type as the entire Let expression. The two variables x,y don't need to have the same type, but x,y and e1 should have the same type:

$$\frac{(x,y) : (T', T''), e1 : (T', T''), e2 : T}{Let(Match(e1, x, y), e2) : T} \tag{1}$$

4 Q 1.7

The expression e1 has to evaluate to some pair (v1,v2). We have to replace occurrences of the pair (x,y) in e2 with (v1,v2).

The resulting e2 expression must evaluate to some value v. In short, we have:

$$\frac{e1 \downarrow (v1, v2), [(v1, v2)/(x, y)]e2 \downarrow v}{letx, y = e1ine2end \downarrow v} \tag{2}$$

5 Q 2.1

For the free variables in *fst* or *snd*, we simply need to compute the free variables of the expression to which these two are applied. In other words,

$$FV(fst(e)) = FV(e)$$
$$FV(snd(e)) = FV(e)$$

This happens because we are looking for the free variables of the entire expression "fst e", not just the free variables of what "fst e" evaluates to (which would be FV(e1) in fst (e1,e2) or FV(e2) in snd (e1,e2)). The same applies to "snd e".

6 Q 2.2

The same applies to substitution as free variables: We want to substitute the variable x with a new expression e' in the entirety of the expression "fst e" or "snd e". The only small nuance is that we now output the initial expression, only with a variable in e substituted. In other words,

$$[e'/x](fst(e)) = fst([e'/x]e)$$
$$[e'/x](snd(e)) = snd([e'/x]e)$$