EECS 662 Spring 2019
Midterm Examination

Sample Solutions

1. **Inference rules** (12 points)**.**
The following inference rules characterize strings of `a`'s and `b`'s.

$$\frac{}{\mathsf{a}\ \mathsf{needs}\ 1} \qquad \frac{}{\mathsf{b}\ \mathsf{needs}\ -1} \qquad \frac{r\ \mathsf{needs}\ m \quad s\ \mathsf{needs}\ n}{rs\ \mathsf{needs}\ m+n}$$

Give derivation trees for the following.

(a) (6 points) `baab` needs 0

$$\frac{\dfrac{\dfrac{}{\mathsf{b}\ \mathsf{needs}\ -1} \quad \dfrac{}{\mathsf{a}\ \mathsf{needs}\ 1}}{\mathsf{ba}\ \mathsf{needs}\ 0} \qquad \dfrac{\dfrac{}{\mathsf{a}\ \mathsf{needs}\ 1} \quad \dfrac{}{\mathsf{b}\ \mathsf{needs}\ -1}}{\mathsf{ab}\ \mathsf{needs}\ 0}}{\mathsf{baab}\ \mathsf{needs}\ 0}$$

(b) (6 points) `abbba` needs -1.

$$\frac{\dfrac{\dfrac{\dfrac{}{\mathsf{a}\ \mathsf{needs}\ 1} \quad \dfrac{}{\mathsf{b}\ \mathsf{needs}\ -1}}{\mathsf{ab}\ \mathsf{needs}\ 0} \quad \dfrac{}{\mathsf{b}\ \mathsf{needs}\ -1}}{\mathsf{abb}\ \mathsf{needs}\ -1} \qquad \dfrac{\dfrac{}{\mathsf{b}\ \mathsf{needs}\ -1} \quad \dfrac{}{\mathsf{a}\ \mathsf{needs}\ 1}}{\mathsf{ba}\ \mathsf{needs}\ 0}}{\mathsf{abbba}\ \mathsf{needs}\ -1}$$

2. **Syntax** (12 points).

Eliminate as many parentheses as possible from the following λ-calculus expressions, *without changing their meaning.*

(a) (3 points) $(\lambda f.(\lambda x.((f\ x)\ x)))$

$\lambda f.\lambda x.f\ x\ x$

(b) (3 points) $\lambda f.(\lambda x.(\lambda y.((f\ (x)) \times (f\ y))))$

$\lambda f.\lambda x.\lambda y.f\ x \times f\ y$

(c) (3 points) $\lambda f.(\lambda g.(\lambda x.(f\ (g\ x))))$

$\lambda f.\lambda g.f\ (g\ x)$

(d) (3 points) $\lambda f.(\lambda x.(f\ (f\ (x))))$

$\lambda f.\lambda x.f\ (f\ x)$

3. **Evaluation** (12 points)**.**
Consider a simple language of arithmetic expressions:

$$t ::= z \mid t + t \mid t \times t \mid \mathsf{if0}\ t\ \mathsf{then}\ t\ \mathsf{else}\ t$$

Evaluation rules for the first three constructs are:

$$\frac{}{z \Downarrow z} \qquad \frac{t_1 \Downarrow z_1 \quad t_2 \Downarrow z_2}{t_1 + t_2 \Downarrow z_1 + z_2} \qquad \frac{t_1 \Downarrow z_1 \quad t_2 \Downarrow z_2}{t_1 \times t_2 \Downarrow z_1 \times z_2}$$

The construct $\mathsf{if0}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3$ should evaluate to the result of t_2 if t_1 evaluates to 0, and t_3 otherwise. Give evaluation rules for this construct.

$$\frac{t_1 \Downarrow 0 \quad t_2 \Downarrow v}{\mathsf{if0}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \Downarrow v} \qquad \frac{t_1 \Downarrow z \quad \overset{\text{t3}}{\cancel{t_2}} \Downarrow v}{\mathsf{if0}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \Downarrow v}\ (z \neq 0)$$

4. **Substitution (I)** (12 points).
Consider a λ-calculus with arithmetic operations:

$$\begin{aligned} t ::= z \mid t + t \mid t \times t \\ \mid x \mid \lambda x.t \mid t\, t \end{aligned}$$

We define substitution for this language as follows:

$$z[v/x] = z \qquad (t_1\; t_2)[v/x] = t_1[v/x]\; t_2[v/x] \qquad (t_1 \odot t_2)[v/x] = t_1[v/x] \odot t_2[v/x],\;\; \odot \in \{+, \times\}$$

$$y[v/x] = \begin{cases} v & \text{if } x = y \\ y & \text{otherwise} \end{cases} \qquad (\lambda y.t)[v/x] = \begin{cases} \lambda y.t & \text{if } x = y \\ \lambda y.t[v/x] & \text{otherwise} \end{cases}$$

subst:相同不变
不同变

Give the result of each of the following substitutions:

(a) (4 points) $(\lambda a.a + b)[5/b]$

$$\lambda a.a + 5$$

(b) (4 points) $(\lambda a.\lambda b.a + b)[5/b]$

$$\lambda a.\lambda b.a + b$$

(c) (4 points) $((\lambda a.a + b)\; a)[5/a]$

$$(\lambda a.a + b)\, 5$$

5. **Substitution (II)** (12 points).
Suppose we extend the previous language with introduction and pattern-matching forms for pairs

$$t ::= \cdots \mid (t, t) \mid \mathsf{let}\ (x, x) = t\ \mathsf{in}\ t$$

with evaluation rules as follow.

$$\frac{t_1 \Downarrow v_1 \quad t_2 \Downarrow v_2}{(t_1, t_2) \Downarrow (v_1, v_2)} \quad \frac{t_1 \Downarrow (v_1, v_2) \quad (t_2[v_1/x_1])[v_2/x_2] \Downarrow v_3}{\mathsf{let}\ (x_1, x_2) = t_1\ \mathsf{in}\ t_2 \Downarrow v_3}$$

Give the corresponding cases of the substitution function for (t_1, t_2) and $\mathsf{let}\ (x_1, x_2) = t_1\ \mathsf{in}\ t_2$.

$$(t_1, t_2)[v/x] = (t_1[v/x], t_2[v/x])$$

$$(\mathsf{let}\ (x_1, x_2) = t_1\ \mathsf{in}\ t_2)[v/x] = \begin{cases} \mathsf{let}\ (x_1, x_2) = t_1[v/x]\ \mathsf{in}\ t_2 & \text{if } x \in \{x_1, x_2\} \\ \mathsf{let}\ (x_1, x_2) = t_1[v/x]\ \mathsf{in}\ t_2[v/x] & \text{otherwise} \end{cases}$$

6. **Evaluation strategies** (12 points).
We can start giving evaluation rules for our language:

$$\frac{}{z \Downarrow z} \qquad \frac{t_1 \Downarrow z_1 \quad t_2 \Downarrow z_2}{t_1 \odot t_2 \Downarrow z_1 \odot z_2} \, (\odot \in \{+, \times\}) \qquad \frac{}{\lambda x.t \Downarrow \lambda x.t}$$

In *call-by-value* evaluation, the argument to a function is evaluated before it is called. In *call-by-name* evaluation, in contrast, the argument to a function is not evaluated before evaluating the function. The difference is illustrated in the evaluation rules below; call-by-value is on the left, and call-by-name is on the right.

$$\frac{t_1 \Downarrow_{\mathsf{cbv}} \lambda x.t \quad t_2 \Downarrow_{\mathsf{cbv}} w \quad t[w/x] \Downarrow_{\mathsf{cbv}} v}{t_1\ t_2 \Downarrow_{\mathsf{cbv}} v} \quad \frac{t_1 \Downarrow_{\mathsf{cbn}} \lambda x.t \quad t[t_2/x] \Downarrow v}{t_1\ t_2 \Downarrow_{\mathsf{cbv}} v}$$

The other rules are identical in call-by-name and call-by-value.

We say that a *step* is one application of the function reduction rule. For example, in either call-by-name or call-by-value, the term $(\lambda a.a+1)\,4$ evaluates to 5 in one step, whereas the term $(\lambda a.\lambda b.a)\,3\,2$ evaluates to 3 in 2 steps. For each of the following terms, give the numbers of steps required to reach the final value under call-by-value and call-by-name interpretations. If a term would never reach a final value, write ∞.

Expression	Steps in cbn	Steps in cbv
$(\lambda f.\lambda a.f\,(f\,x))\,(\lambda b.b)\,3$	4	4
$(\lambda a.\lambda b.b)\,((\lambda f.f\,f)\,(\lambda f.f\,f))\,3$	2	∞
$(\lambda a.a+a)\,((\lambda b.b)\,3)$	3	2

一个lambda 一个step
call by value 如果出现相同的lambda，将会是无限循环

7. **Fixed points** (12 points).

The *fixed point* construct captures recursive definition of values:

$$t ::= \cdots \mid \mathsf{fix}\, t$$

with the evaluation rule

$$\frac{}{\mathsf{fix}\, t \Downarrow \lambda x.t\, (\mathsf{fix}\, t)\, x}$$

Rewrite the following recursive definitions to used the fixed point construct instead.

(a) (6 points) $\mathit{fib} = \lambda n.\mathsf{if}\ n \leq 1\ \mathsf{then}\ n\ \mathsf{else}\ \mathit{fib}\,(n-1) + \mathit{fib}\,(n-2)$

$$\boxed{\mathsf{fix}\,(\lambda \mathit{fib}.\lambda n.\mathsf{if}\ n \leq 1\ \mathsf{then}\ n\ \mathsf{else}\ \mathit{fib}\,(n-1) + \mathit{fib}\,(n-2))}$$

(b) (6 points) $\mathit{odd} = \lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ \mathit{False}\ \mathsf{else}\ \mathsf{if}\ n = 1\ \mathsf{then}\ \mathit{True}\ \mathsf{else}\ \neg(\mathit{odd}\,(n-1))$

$$\boxed{\mathsf{fix}\,(\lambda \mathit{odd}.\lambda n.\mathsf{if}\ n = 1\ \mathsf{then}\ \mathit{False}\ \mathsf{else}\ \mathsf{if}\ n = 1\ \mathsf{then}\ \mathit{True}\ \mathsf{else}\ \neg(\mathit{odd}\,(n-1)))}$$