COP5025 Midterm Exam – Spring 2001

Name: ______________________________ (Please print)

You can put all answers on these sheets. Use additional sheets when necessary. Always show how you derived your answers (this is required for full credit and helpful for partial credit). You can collect 100 points in total for this exam. A bonus question is included for an additional 15 points.

1. (10 points) The genetic code DNA of a living being is encoded in chromosomes, part of every living cell. Each chromosome is a sequence of nucleotides A, C, T, or G. Messenger RNA (mRNA) is a replication of the DNA sequence, and it used in the formation of amino acids which are triplets over A, C, U, and G. Give a regular expression for the triplets for the amino acid *Arginine*, described by the language $L = \{\text{CGU,CGC,CGA,CGG,AGA,AGG}\}$ over the alpahabet $\Sigma = \{A, C, U, G\}$. Make this single regular expression for Arginine *as compact as possible* by using the *distribution rule*:

   $$r_1\ r_2 \,|\, r_1\ r_3 = r_1\ (r_2 \,|\, r_3)$$

   where r_1, r_2, and r_3 are regular expressions (concatenation is left associative, so r_1 can be a concatenation).

   Answer: $CG(U|C|A|G)|AG(A|G)$

2. (10 points) Prove that the following equivalence holds for regular expression r:

   $$r\ (\epsilon \,|\, \emptyset) = r$$

   Use the mapping $\mathcal{D} : RE \to Lang$ defined on lecture note 16 and the language operations defined on lecture note 14. Note that $s \cdot \epsilon = s$

   Answer:

   $$\begin{aligned}
   \mathcal{D}[\![r\ (\epsilon \,|\, \emptyset)]\!] &= \{x \cdot y \,|\, x \in \mathcal{D}[\![r]\!] \wedge y \in \mathcal{D}[\![\epsilon \,|\, \emptyset]\!]\} \\
   &= \{x \cdot y \,|\, x \in \mathcal{D}[\![r]\!] \wedge y \in \mathcal{D}[\![\epsilon]\!] \cup \mathcal{D}[\![\emptyset]\!]\} \\
   &= \{x \cdot y \,|\, x \in \mathcal{D}[\![r]\!] \wedge y \in \{\epsilon\} \cup \emptyset\} \\
   &= \{x \cdot y \,|\, x \in \mathcal{D}[\![r]\!] \wedge y = \epsilon\} \\
   &= \{x \cdot \epsilon \,|\, x \in \mathcal{D}[\![r]\!]\} \\
   &= \{x \,|\, x \in \mathcal{D}[\![r]\!]\} \\
   &= \mathcal{D}[\![r]\!]
   \end{aligned}$$

   Or you can use the identities on note 16:

   $$\begin{aligned}
   r\ (\epsilon \,|\, \emptyset) &= r\,\epsilon \,|\, r\,\emptyset \\
   &= r \,|\, \emptyset \\
   &= r
   \end{aligned}$$

3. Suppose a robot can move in four directions: `up`, `down`, `left`, and `right`. The robot can also `get` an item and `put` an item down (the robot can hold more than one item). The following attribute grammar describes the robot's motions and get/put operations:

$robot_1$	::= $robot_2$ $move$	$robot_1$.x:=$robot_2$.x+$move$.x; $robot_1$.y:=$robot_2$.y+$move$.y; $robot_1$.n:=$robot_2$.n
$robot_1$	::= $robot_2$ $oper$	$robot_1$.x:=$robot_2$.x; $robot_1$.y:=$robot_2$.y; $robot_1$.n:=$robot_2$.n+$oper$.n
$robot$	::= $move$	$robot$.x:=$move$.x; $robot$.y:=$move$.y; $robot$.n:=0
$robot$	::= $oper$	$robot$.x:=0; $robot$.y:=0; $robot$.n:=$oper$.n
$move$	::= `up`	$move$.x:=0; $move$.y:=1
$move$	::= `down`	$move$.x:=0; $move$.y:=-1
$move$	::= `left`	$move$.x:=-1; $move$.y:=0
$move$	::= `right`	$move$.x:=1; $move$.y:=0
$oper$	::= `get`	$oper$.n:=1
$oper$	::= `put`	$oper$.n:=-1

An example parse tree for `get left up` is:

```
                                   robot.x=1
                                   robot.y=2
                                   robot.n=1
                                    /    |
                                   /   oper.n=-1
                            robot.x=1    |
                            robot.y=2  put
                            robot.n=2
                             /    |
                            /   move.x=0
                     robot.x=1  move.y=1
                     robot.y=1    |
                     robot.n=2  up
                      /    |
                     /   oper.n=1
              robot.x=1    |
              robot.y=1  get
              robot.n=1
               /    |
              /   move.x=1
       robot.x=0  move.y=0
       robot.y=1    |
       robot.n=1  right
        /    |
       /   move.x=0
robot.x=0  move.y=1
robot.y=0    |
robot.n=1  up
  |
 oper.n=1
  |
 get
```

(a) (10 points) Draw a parse tree for the string `get up right get up put`.

(b) (15 points) Annotate the parse tree. What is the final (x,y) position of the robot? How many items n does it carry at the end?

Answer: The final position is $(1, 2)$ and the robot caries $n = 1$ items at the end.

4. (20 points) The typing rule for the `let`-construct in Mini-ML is given by

$$\frac{A \vdash e_1 : \sigma \qquad A[x \mapsto \sigma] \vdash e_2 : \tau}{A \vdash \texttt{let } x = e_1 \texttt{ in } e_2 \texttt{ end} : \tau}$$

Infer the type of the expression

`let f = (fn n => 1) in f x end`

starting with the type assignments $A = \{\langle 1, \texttt{int}\rangle, \langle \texttt{x}, \texttt{bool}\rangle\}$. Show the entire Post system derivation.

Answer: will be demonstrated in class.

5. (10 points) Find the free variables of the lambda expressions

- $f\ x$ *Answer:* $\{x\}$
- $(\lambda\, x\, .\, x)\ \ (f\ x)$ *Answer:* $\{x\}$
- $\lambda\, x\, .\, y$ *Answer:* $\{y\}$
- $\lambda\, x\, .\, (\lambda\, x\, .\, x)\ \ y$ *Answer:* $\{y\}$
- $(\lambda\, x\, .\, \lambda\, y\, .\, x)\ \ (\lambda\, x\, .\, y)$ *Answer:* $\{y\}$

6. (15 points) Use *normal order beta reduction* (NOR) to reduce the following lambda expressions to *weak head normal form* (WHNF). Also show the beta normal forms by reducing the expressions further. Show all reduction steps. Be careful to avoid the name clash problem!

(a) $(\lambda\, x\, .\, \lambda\, y\, .\, x)\ \ (f\ y)$
Answer: $(\lambda\, x\, .\, \lambda\, y\, .\, x)\ \ (f\ y) \rightarrow_\beta \lambda\, v\, .\, f\ y$ (WHNF and NF)

(b) $\lambda\, x\, .\, \lambda\, y\, .\, (\lambda\, z\, .\, x)\ y$
Answer: $\lambda\, x\, .\, \lambda\, y\, .\, (\lambda\, z\, .\, x)\ y$ (WHNF)
Reducing to NF: $\lambda\, x\, .\, \lambda\, y\, .\, (\lambda\, z\, .\, x)\ y \rightarrow_\beta \lambda\, x\, .\, \lambda\, y\, .\, x$

(c) $(\lambda\, x\, .\, \lambda\, y\, .\, \lambda\, z\, .\, x\ z\ \ (y\ z))\ (\lambda\, x\, .\, \lambda\, y\, .\, x)\ (\lambda\, x\, .\, x)$
Answer:

$$\begin{aligned} & (\lambda\, x\, .\, \lambda\, y\, .\, \lambda\, z\, .\, x\ z\ \ (y\ z))\ (\lambda\, x\, .\, \lambda\, y\, .\, x)\ (\lambda\, x\, .\, x) \\ \rightarrow_\beta &\quad (\lambda\, y\, .\, \lambda\, z\, .\, (\lambda\, x\, .\, \lambda\, y\, .\, x)\ \ z\ \ (y\ z))\ (\lambda\, x\, .\, x) \\ \rightarrow_\beta &\quad \lambda\, z\, .\, (\lambda\, x\, .\, \lambda\, y\, .\, x)\ \ z\ \ ((\lambda\, x\, .\, x)\ \ z) \qquad \text{(WHNF)} \end{aligned}$$

Reducing further to NF: $\rightarrow_\beta \lambda\, z\, .\, (\lambda\, y\, .\, z)\ \ ((\lambda\, x\, .\, x)\ \ z) \rightarrow_\beta \lambda\, z\, .\, z$

7. (10 points) Apply combinator reduction to reduce $\mathbf{S}\ \ (\mathbf{S}\ \mathbf{K})\ \ \mathbf{I}\ \mathbf{K}$, where

$$\begin{aligned} \mathbf{S}\ x\ y\ z &\ >_C\ x\ z\ \ (y\ z) \\ \mathbf{K}\ x\ y &\ >_C\ x \\ \mathbf{I}\ x &\ >_C\ x \end{aligned}$$

for all expressions x, y, and z. Recall that application is left associative, so $\mathbf{S}\ a\ b\ c$ is shorthand for $(((\mathbf{S}\ a)\ b)\ c)$. Show all reduction steps.

Answer:

$$\begin{aligned} \mathbf{S}\ \ (\mathbf{S}\ \mathbf{K})\ \ \mathbf{I}\ \mathbf{K} &\ \rightarrow_C\ (\mathbf{S}\ \mathbf{K})\ \mathbf{K}\ \ (\mathbf{I}\ \mathbf{K}) \\ &\ =\ \mathbf{S}\ \mathbf{K}\ \mathbf{K}\ \ (\mathbf{I}\ \mathbf{K}) \\ &\ \rightarrow_C\ \mathbf{K}\ \ (\mathbf{I}\ \mathbf{K})\ \ (\mathbf{K}\ \ (\mathbf{I}\ \mathbf{K})) \\ &\ \rightarrow_C\ \mathbf{I}\ \mathbf{K} \\ &\ \rightarrow_C\ \mathbf{K} \end{aligned}$$

8. **Bonus question.** (15 points)

Suppose we define

$$\begin{aligned} \text{TRUE} &:= \mathbf{K} \\ \text{NOT} &:= \mathbf{C} \\ \text{FALSE} &:= \text{NOT TRUE} = \mathbf{C}\ \mathbf{K} \end{aligned}$$

where

$$\mathbf{C}\ x\ y\ z >_C x\ z\ y$$

(a) (5 points) Show that we can define a conditional expression by

$$(\textbf{if } x \textbf{ then } y \textbf{ else } z) := x\ y\ z$$

That is, show that

$$(\textbf{if } \text{TRUE} \textbf{ then } y \textbf{ else } z) = \text{TRUE}\ y\ z \xrightarrow{\star}_C y$$

and

$$(\textbf{if } \text{FALSE} \textbf{ then } y \textbf{ else } z) = \text{FALSE}\ y\ z \xrightarrow{\star}_C z$$

Answer:

$$\begin{aligned} \text{TRUE}\ y\ z &= \mathbf{K}\ y\ z \\ &\rightarrow_C y \end{aligned}$$

and

$$\begin{aligned} \text{FALSE}\ y\ z &= \mathbf{C}\ \mathbf{K}\ y\ z \\ &\rightarrow_C \mathbf{K}\ z\ y \\ &\rightarrow_C z \end{aligned}$$

(b) (5 points) Let's define a logical OR operation

$$\text{OR}\ x\ y := \textbf{if } x \textbf{ then } \text{TRUE} \textbf{ else } y$$

We can write this as

$$\text{OR} := \lambda x\,.\,\lambda y\,.\,x\ \text{TRUE}\ y$$

Now, convert $\lambda x\,.\,\lambda y\,.\,x$ TRUE y to a combinator term using Curry's optimized bracket abstraction:

$$\begin{aligned} [x]y &:= \begin{cases} \mathbf{I} & \text{if } y = x \\ \mathbf{K}\,y & \text{otherwise} \end{cases} \\ [x]f &:= \mathbf{K}\,f \quad \text{for any function symbol or combinator } f \\ [x](L\ x) &:= L \quad \text{if } x \notin FV[L] \\ [x](L_1\ L_2) &:= \mathbf{K}\ (L_1\ L_2) \quad \text{if } x \notin FV[L_1] \wedge x \notin FV[L_2] \\ [x](L_1\ L_2) &:= \mathbf{C}\ ([x]L_1)\ L_2 \quad \text{if } x \in FV[L_1] \wedge x \notin FV[L_2] \\ [x](L_1\ L_2) &:= \mathbf{B}\,L_1\ ([x]L_2) \quad \text{if } x \notin FV[L_1] \wedge x \in FV[L_2] \\ [x](L_1\ L_2) &:= \mathbf{S}\ ([x]L_1)\ ([x]L_2) \quad \text{if } x \in FV[L_1] \wedge x \in FV[L_2] \end{aligned}$$

(The combinator term that you get represents OR.)

Answer:

$$
\begin{aligned}
\text{OR} &= \mathcal{N}[\![\lambda x\,.\,\lambda y\,.\,x\ \mathbf{K}\ y]\!] \\
&= [x]\mathcal{N}[\![\lambda y\,.\,x\ \mathbf{K}\ y]\!] \\
&= [x]([y]\mathcal{N}[\![x\ \mathbf{K}\ y]\!]) \\
&= [x]([y](\mathcal{N}[\![x\ \mathbf{K}]\!]\ \mathcal{N}[\![y]\!])) \\
&= [x]([y](\mathcal{N}[\![x]\!]\ \mathcal{N}[\![\mathbf{K}]\!]\ \mathcal{N}[\![y]\!])) \\
&= [x]([y](x\ \mathbf{K}\ y)) \\
&= [x](x\ \mathbf{K}) \qquad \text{with third rule} \\
&= \mathbf{C}\ \ ([x]x)\ \ \mathbf{K} \\
&= \mathbf{C\ I\ K} \qquad \text{with first rule}
\end{aligned}
$$

(c) (5 points) With the above, show that

$$\text{OR FALSE TRUE} \equiv_C \text{TRUE}$$

Answer:

$$
\begin{aligned}
\mathbf{C\ I\ K}\ \ (\mathbf{C\ K})\ \ \mathbf{K} &\rightarrow_C \mathbf{I}\ \ (\mathbf{C\ K})\ \ \mathbf{K\ K} \\
&\rightarrow_C \mathbf{C\ K\ K\ K} \\
&\rightarrow_C \mathbf{K\ K\ K} \\
&\rightarrow_C \mathbf{K}
\end{aligned}
$$