CMSC330 - Organization of Programming Languages
Spring 2023 - Exam 2 Solution

CMSC330 Course Staff
University of Maryland
Department of Computer Science

Name: ______________________________

UID: ______________________________

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination

Signature: ______________________________

Ground Rules

- You may use anything on the accompanying reference sheet anywhere on this exam
- Please write legibly. **If we cannot read your answer you will not receive credit**
- You may not leave the room or hand in your exam within the last 10 minutes of the exam
- The last page is blank and scratch work can be done there.
- If anything is unclear, ask a proctor. If you are still confused, write down your assumptions in the margin

Question	Points
Q1	10
Q2	18
Q3	10
Q4	12
Q5	15
Q6	15
Total	80

Problem 1: Language Concepts

[Total 10 pts]

An improper garbage collector can cause security vulnerabilities [✓ **True**] **False**

Having things stay in memory for too long is a security vulnerability

Modern Languages use a combination of Reference Counting, Mark and Sweep and Stop and Copy [✓ **True**] **False**

Stated in Lecture, more effiiceint to do this

Lambda Calculus Expressions can be converted to Finite State Machines **True** [✓ **False**]

Lambda Calculus cannot be expressed as a FSM

The relation of FSM to Regex is bijective (1 to 1) **True** [✓ **False**]

Some NFAs represent the same regex

Eager and Lazy Evaluation will always give the same result **True** [✓ **False**]

Consider cases of infinite reduction: $(\lambda x.a)((\lambda x.xx)(\lambda y.yy))$

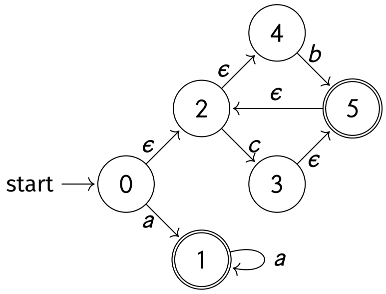
Problem 2: Finite State Machines

[Total 18 pts]

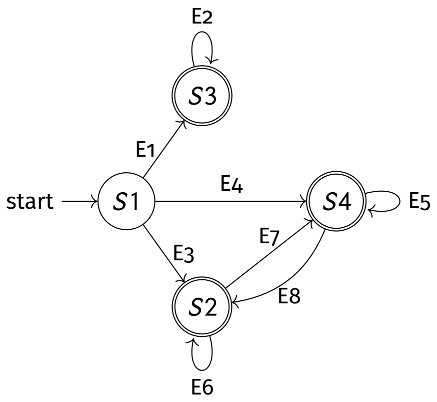
(a) Using the subset algorithm, convert the following NFA to a DFA, and fill in the blanks appropriately matching the DFA provided with the right nodes and transitions. Only the blanks will be graded. [12 pts]

NFA:

Scratch Space (if needed)



DFA:



So since state $S4$ and $S2$ can be swapped, any transitions to them can also be interchanged.

S1: 0,2,4
S2: 2,4,5 | 2,3,4,5
S3: 1
S4: 2,3,4,5 | 2,4,5

E1: a
E2: a
E3: b | c
E4: c | b

E5: c | b
E6: b | c
E7: c | b
E8: b | c

[3 pts] (b) Write a regex to describe the language of the above NFA

$(a+)|(b|c)+$

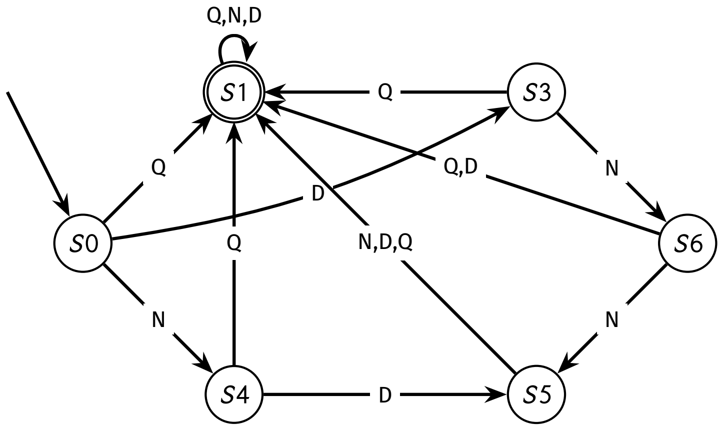
States 0 and 1 represent $a+$, States $2,4,5$ represent $b+$, States $2,3,5$ represent $c+$. So together, $2,3,4,5$ represents $(c|b)+$.

[3 pts] (c) Vending Machine Fun

Suppose there is a vending machine which takes in quarters (Q), dimes (D) and nickles (N). Consider the following actions you can perform when interacting with the vending machine:

Action N: Insert a Nickle Action D: Insert a Dime Action Q: Insert a Quarter

The price of each item is \$0.25. However, the FSM for the machine was leaked and turns out you can pay less than \$0.25 per item. List out the operations you want to perform to pay less than \$0.25. For example, if you wanted to put in 2 quarters, followed by 1 dime, followed by 3 nickles, your answer should be `Q,Q,D,N,N,N`.



N,D,N is \$0.20 ($S0 \longrightarrow S4 \longrightarrow S5 \longrightarrow S1$)

Problem 3: CFGs

[Total 10 pts]

Consider the following Grammars:

Grammar 1

$$\begin{aligned} S \longrightarrow\ & aSb \\ |\ & aaSb \\ |\ & aaaSb \\ |\ & \epsilon \end{aligned}$$

Grammar 2

$$\begin{aligned} S &\longrightarrow AAASB \mid \epsilon \\ A &\longrightarrow a \mid \epsilon \\ B &\longrightarrow b \end{aligned}$$

Grammar 3

$$\begin{aligned} S &\longrightarrow ASB \\ A &\longrightarrow aA \mid \epsilon \\ B &\longrightarrow bbbB \mid \epsilon \end{aligned}$$

(a) Which of the following grammars describe strings of $a^x b^y$, $x < 3y$? Select all that apply.

[2 pts]

Grammar 1 Grammar 2 Grammar 3 [✓ None]

Not possible because if $y = 0$, then x has to be negative. Grammar 1 and Grammar 2 describe the same thing: $x \leq 3y$. The third does not terminate.

(b) Prove that Grammar 2 is ambiguous

[3 pts]

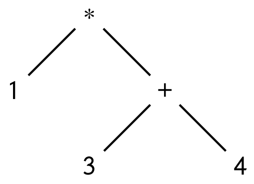
$S \rightarrow AAASB \rightarrow AASB \rightarrow ASB \rightarrow aSB \rightarrow aB \rightarrow ab$

$S \rightarrow AAASB \rightarrow aAASB \rightarrow aASB \rightarrow aSB \rightarrow aB \rightarrow ab$

(c) Draw the abstract syntax tree that would be generated by parsing the following string with the given CFG using a leftmost derivation.

[5 pts]

String: "1 * 2 + 3"
CFG:
S -> M * S | M
M -> M + N | N
N -> 1 | 2 | 3 | (N), where n is any number



[Total 12 pts] **Problem 4: Operational Semantics**

Consider the following rules for LOLCODE, using OCaml as the Metalanguage:

Rule 1: $\dfrac{}{\text{WIN} \longrightarrow \text{WIN}}$

Rule 2: $\dfrac{}{\text{FAIL} \longrightarrow \text{FAIL}}$

Rule 3: $\dfrac{A;\ e_1 \Rightarrow v_1 \qquad A;\ e_2 \Rightarrow v_2 \qquad v_1\ <>\ v_2}{A;\ \text{DIFFRINT } e_1 \text{ AN } e_2 \Rightarrow \text{WIN}}$

Rule 4: $\dfrac{A;\ e_1 \Rightarrow v_1 \qquad A;\ e_2 \Rightarrow v_2 \qquad v_1\ =\ v_2}{A;\ \text{DIFFRINT } e_1 \text{ AN } e_2 \Rightarrow \text{FAIL}}$

Rule 5: $\dfrac{A, \text{x} : v\ (\text{x})\ =\ v}{A, \text{x} : v;\ \text{x} \Rightarrow v}$

Rule 6: $\dfrac{A;\ e_1 \Rightarrow v_1 \qquad A, \text{x} : v_1;\ e_2 \Rightarrow v_2}{A;\ \text{HAS A x ITZ } e_1 \ \backslash\text{n } e_2 \Rightarrow v_2}$

Rule 7: $\dfrac{A;\ e_1 \Rightarrow v_1 \qquad A;\ e_2 \Rightarrow v_2 \qquad v_3 = \text{if } v_1 > v_2 \text{ then } v_1 \text{ else } v_2}{A;\ \text{BIGGR OF } e_1 \text{AN } e_2 \Rightarrow v_3}$

Rule 8: $\dfrac{}{A; n \longrightarrow n}$

[4 pts] (a) What are the axioms in this language? Select all the apply.

[✓ Rule 1] [✓ Rule 2] Rule 3 Rule 4 Rule 5 Rule 6 Rule 7 [✓ Rule 8] none

[8 pts] (b) Complete the opsem proof for the following program:

HAS A x ITZ 7 \n DIFFRINT 2 AN (BIGGR OF 2 AN x) => WIN

$$\dfrac{\dfrac{}{\boxed{1}} \qquad \dfrac{\dfrac{}{\boxed{3}} \qquad \dfrac{\dfrac{}{A, x : 7; 2 \Rightarrow 2} \qquad \dfrac{\boxed{8}}{A, x : 7; x \Rightarrow 7} \qquad 7 = \boxed{7}}{A, x : 7; \boxed{4} \Rightarrow \boxed{5}} \qquad \boxed{6}}{A, x : 7;\ \text{DIFFRINT 2 AN } \boxed{2} \Rightarrow \text{WIN}}}{A; \text{HAS A x ITZ 7 } \backslash\text{n DIFFRINT 2 AN (BIGGR OF 2 AN x)} \Rightarrow \text{WIN}}$$

Blank 1: $A; 7 \Rightarrow 7$

Blank 2: BIGGR OF 2 AN x

Blank 3: $A, x : 7; 2 \Rightarrow 2$

Blank 4: BIGGR OF 2 AN x

Blank 5: 7

Blank 6: 2 <> 7

Blank 7: if 2 $>$ 7 then 2 else 7

Blank 8: $A, x : 7(x) = 7$

Problem 5: Lambda Calculus

[Total 15 pts]

For the following questions perform a single β-reduction using lazy (call by name) evaluation on the outermost expression. If you cannot reduce it, write **Beta Normal Form**. You may **not** α-convert your final answer.

(a) $(\lambda x.x\lambda y.xy)(y(\lambda x.yx))$ [2 pts]

$(y(\lambda x.yx))\lambda y.(y(\lambda x.yx))y$ - We use $(y(\lambda x.yx))$ as input to $(\lambda x.x\lambda y.xy)$

(b) $(\lambda x.\lambda x.xx)((\lambda x.yx)((\lambda a.aa)b))$ [2 pts]

$\lambda x.xx$ - We use $((\lambda x.yx)((\lambda a.aa)b))$ as input to $(\lambda x.\lambda x.xx)$

For the following questions perform a single β-reduction using Eager (call by value) evaluation on the outermost expression. If you cannot reduce it, write **Beta Normal Form**. You may **not** α-convert your final answer.

(c) $(\lambda x.x\lambda y.xy)(y(\lambda x.yx))$ [2 pts]

$(y(\lambda x.yx))\lambda y.(y(\lambda x.yx))y$ - we still need to beta reduce
but since the argument cannot be reduced further, we just reduce the outermost expression. Same as part (a)

(d) $(\lambda x.\lambda x.xx)((\lambda x.yx)((\lambda a.aa)b))$ [2 pts]

$(\lambda x.\lambda x.xx)((\lambda x.yx)(bb))$ or $(\lambda x.\lambda x.xx)(y((\lambda a.aa)b))$. We did not talk about what to do here

(e) Convert the following to Beta Normal Form: $(\lambda x.(\lambda y.xa)b)(\lambda x.ax)$ [3 pts]

$\lambda x.ax$ $c\ d$ $b\ a$ [✓ $a\ a$] can't reduce infinite recursion None

Consider the following lambda calculus bindings:
true = $\lambda x.\lambda y.x$
false = $\lambda x.\lambda y.y$
if e1 then e2 else e3 = e1 e2 e3

(f) Encode the following expression: `if false then false else true` [4 pts]

$(\lambda x.\lambda y.y)(\lambda x.\lambda y.y)(\lambda x.\lambda y.x)$

[Total 15 pts]

Problem 6: Lexing, Parsing, Evaluation

Consider the following modified Math-ew from lecture:

$$\begin{aligned} E &\Rightarrow\ + E\ E \mid *\ E\ E \mid sq\ E \mid exp\ E\ E \mid and\ E\ E \mid or\ E\ E \mid N \\ N &\Rightarrow\ 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid true \mid false \end{aligned}$$

You may assume that the behaviour is the same as Ocaml.

[5 pts] (a) Lexing

Which of the following phrases would fail the lexing stage for the Math-ew Language? Please bubble in the circle

Ⓐ `2 * 3 sq 2 3` Ⓑ `4 ^ 5` Ⓒ `- + 1 23`

Ⓓ `exp -2 5` Ⓔ `5 exp 2 + 6` Ⓕ `* 2 and true false`

Ⓖ `and true or false false` Ⓗ `false true` Ⓘ `true and false or true`

B, C, D: Basically, which phrases have symbols not in the grammar?

[5 pts] (b) Parsing

Which of the following phrases would fail the parsing stage for the Math-ew Language? If it failed the lexing phase, **do not** mark it.

Ⓐ `2 * 3 sq 2 3` Ⓑ `4 ^ 5` Ⓒ `- + 1 23`

Ⓓ `exp -2 5` Ⓔ `5 exp 2 + 6` Ⓕ `* 2 and true false`

Ⓖ `and true or false false` Ⓗ `false true` Ⓘ `true and false or true`

A,E,H,I: Basically, which phrases are grammatically incorrect?

[5 pts] (c) Evaluation

Which of the following phrases would fail the evaluator stage for the Math-ew Language? If it failed the lexing or parsing phase, **do not** mark it.

Ⓐ `2 * 3 sq 2 3` Ⓑ `4 ^ 5` Ⓒ `- + 1 23`

Ⓓ `exp -2 5` Ⓔ `5 exp 2 + 6` Ⓕ `* 2 and true false`

Ⓖ `and true or false false` Ⓗ `false true` Ⓘ `true and false or true`

F: Basically, which phrases don't make sense? The only 2 left are F anf G. Since we said behaviour is same as Ocaml, we can definitely `or` and and booleans of true and false. We cannot however, multiply 2 and the result of and-ing `true` and `false`

You can use this page for scratch work: