Your name ___________________________

Harvey Mudd College
**CS 81 Mid-Term Exam: Sample Solutions**
Fall semester, 2006

Four sections on seven pages, 12 problems
100 Points
Closed book

## Instructions

The exam is closed book. Do not refer to materials during the examination, other than a 2-sided 8.5 x 11” crib-sheet as described previously. Do not use a computer, calculator, or pda (personal digital assistant).

Please provide answers to the problems directly on these pages. Some problem pages are printed on the back to allow a 2-page facing space for your answer.

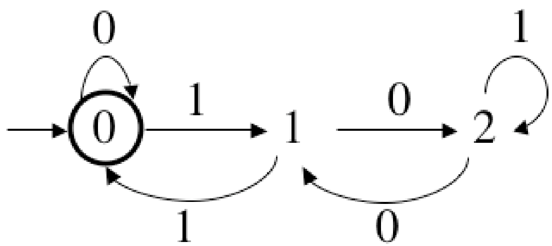
The exam has a time limit of 75 minutes. While you might not finish, work so as to maximize total points, avoiding getting stuck on one problem at the expense of others. It is suggested that you look over the exam first to get an idea of how to apportion your time.

Please be careful.

1. [10 points]

Derive a regular expression for the set of all strings of 0's and 1's representing a multiple of 3 as a binary numeral.

Hint: The following DFA was shown in the text to accept this set of strings:




**Answer**: (0 + 1 ( 0 1* 0)* 1)*

The answer can be read off from the diagram: An path in the language must go from state 0 to state 0. This can be done by any number of iterations of the Small loop from 0 to 0 or the Larger nested loops: (S + L)*. Here S is 0 and L is 1 P* 1 where P is the mid-level Pair of nested loops. P is 0 1* 0. So we have L = 1 (0 1* 0), and overall (0 + 1 ( 0 1* 0)* 1)*.

We could also work this out using successive state elimination. [I will do this in class if asked.]

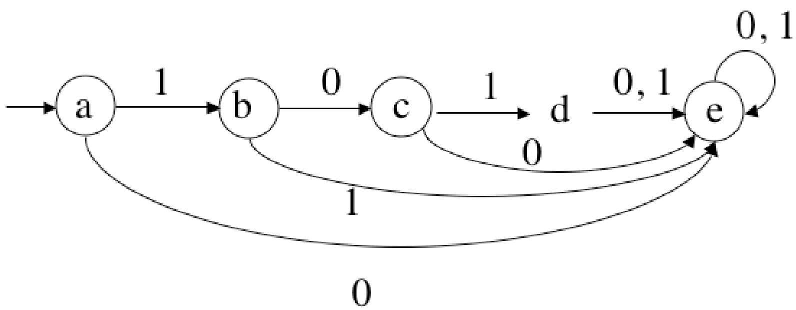
We could also use the equivalent regular expression: (0* (1 ( 0 1* 0)* 1)* )*, since (A + B)* = (A*B*)*.

[20 points]

For each language below:

- If the language is regular, construct the minimal DFA for it.
- If the language is not regular, describe the abstract states of the language and how they are connected.

a. The set of all strings in {0, 1}* other than 101. **This set is regular.**

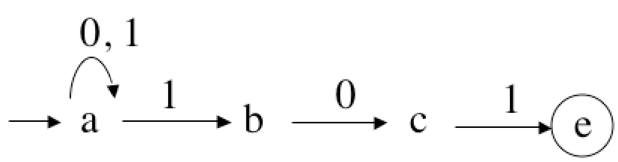



To see that this DFA is minimal, compute partitions:

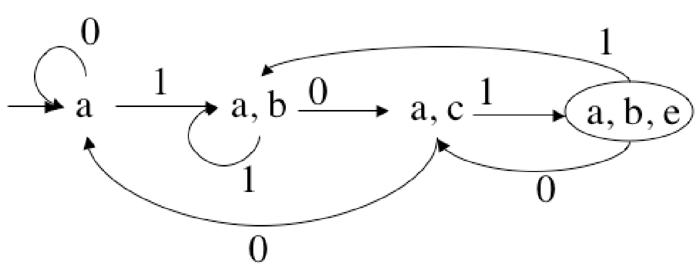
$P_0$ = {{a, b, c, e}, {d}}
$P_1$ = {{a, b, e}, {c}, {d}}
$P_2$ = {{a, e}, {b}, {c}, {d}}
$P_3$ = {{a}, {b}, {c}, {d}, {e}}

b. The set of all strings in {0, 1}* ending in 101. **This set is regular.**

First construct an NFA accepting this set:




Then use the subset construction to convert it to a DFA:



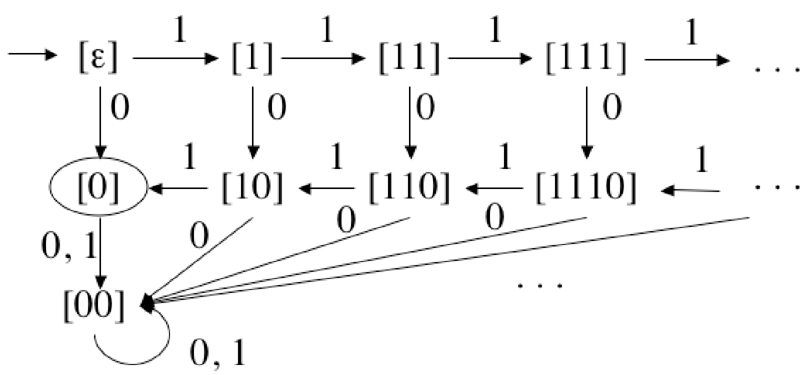

Similar to part a, this can be seen to be minimal.

c. The set {x 0 x | x ∈ {1}*}.

**This set is not regular**. The abstract states are shown as nodes in the diagram below.

Each prefix of a string in the language is not equivalent to any other, as it can be extended to a string in the language by any member of a set of strings unique to the particular prefix. (For example, the set of extensions for 11 is {011, 10111, 1101111, …}.

Any deviation from a prefix of a string in the language goes to the class of 00, which is rejecting and from which no accepting state can be reached.



2. [40 points]

For each language below:

a. If the language is regular, describe the operation of a DFA or NFA accepting the language.

b. If the language is not regular, but is context-free, describe the operation of a (possibly non-deterministic) pushdown acceptor accepting the language.

c. If the language is not context-free, give a convincing argument for this fact.

Languages ($x^r$ designates the reverse of string x and |x| is the length of x.):

a. $A = \{x\, x^r \mid x \in \{0, 1\}^*\}^*$ [note the outermost *]

   **A is context-free but not regular**. A PDA for A repeats the following process an arbitrary number of times: Read input, pushing each symbol read on the stack. At some point, guess that it is time to start popping the stack. Each symbol read is paired with an equal one on the stack. When the initial stack symbol is on top, return to the pushing phase. Whenever the initial stack symbol is on top, the PDA can pop it using no input, and accept what has been seen so far.

b. $B = \{x\, x^r\, x\, x^r \mid x \in \{0, 1\}^*\}$

   **B is not context free**. This can be demonstrated using the pumping lemma. Assume B is context-free. Let k = pump(B). Consider $x = 0^k1^k$. The corresponding element of B is $0^k1^k1^k0^k0^k1^k1^k0^k$. If we identify this string as uvwxy, where $|vwx| \leq k$, we note that vwx must be of the form 0*1* or 1*0*. When vx is pumped, for example $v^2yx^2$, the resulting string is not in B, which provides a contradiction to the assumption.

c. $C = \{x\, y\, y^r \mid x \in \{0, 1\}^*, y \in \{0, 1\}^*, |x| <1000\}$

   **C is context free, but not regular**. A PDA accepting C operates as follows: It begins reading input symbols. Somewhere within the first 999 characters, it makes a guess that it is time to start reading y. It pushes the symbols it thinks form y onto the stack as they are read. Then at some point, it guesses that y is finished and $y^r$ is beginning. Accordingly it begins matching the remaining symbols against symbols on the stack. When the initial stack symbol appears, it pops it leaving an empty stack, accepting what is read so far.

d. $D = \{x\, y\, y^r \mid x \in \{0, 1\}^*, y \in \{0, 1\}^*, |y| <1000\}$

   **D is regular**. An NFA operates as follows: It begins reading input symbols, by looping on its initial state, which is also final. At some point, it guesses that y is starting. It remembers all symbols of y in its states. Since there are at most 999 of them, this is possible. At some point, it guesses that $y^r$ is starting, so it begins making transitions through a set of states that parallel those remembering y. When and if it reaches the state corresponding to $y^r$, it accepts.

4. [30 points]

For each logical expression below:

a. Determine whether or not the expression is a tautology using any method (including inspection, but be careful).

b. If it is a tautology, prove it using natural deduction.

c. If it is not a tautology, give a counterexample (in the form of a valuation).

E: $((p \wedge r) \to (q \wedge s)) \to ((p \to q) \wedge (r \to s))$

F: $((p \to q) \wedge (r \to s)) \to ((p \wedge r) \to (q \wedge s))$

G: $(p \to q) \to (\neg p \to \neg q)$

H : $(p \to q) \vee (q \to r)$

---

**E**: $((p \wedge r) \to (q \wedge s)) \to ((p \to q) \wedge (r \to s))$ **is not a tautology**, by the valuation: $v(p) = 1, v(r) = 0, v(q) = 0, v(s) = 1$. Then $v(p \wedge r) = 0$, so $v((p \wedge r) \to (q \wedge s)) = 1$, but $v(p \to q) = 0$, so $v((p \to q) \wedge (r \to s)) = 0$, giving an overall value of 0.

---

**F**: **is a tautology**, by the following proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\,(p \to q) \wedge (r \to s)\,]_2}{(p \to q)}(\wedge E) \quad \dfrac{[p \wedge r\,]_1}{p}(\wedge E)}{q}(\to E) \qquad \dfrac{\dfrac{[\,(p \to q) \wedge (r \to s)\,]_2}{(r \to s)}(\wedge E) \quad \dfrac{[p \wedge r\,]_1}{r}(\wedge E)}{s}(\to E)}{q \wedge s}(\wedge I)}{(p \wedge r) \to (q \wedge s)}(\to I_1)}{((p \to q) \wedge (r \to s)) \to ((p \wedge r) \to (q \wedge s))}(\to I_2)$$

---

**G**: $(p \to q) \to (\neg p \to \neg q)$ is **not a tautology**, by the valuation: $v(p) = 0, v(q) = 1$. Then $v(p \to q) = 1$, but $v(\neg p \to \neg q) = 0$.

---

**H**: $(p \to q) \vee (q \to r)$ **is a tautology**, by the following proof:

| | | |
|---|---|---|
| 1. | $q \vee \neg q$ | LEM (Law of the Excluded Middle, van D, p 52) |
| 2. | $q$ | Assumption |
| 3. | $p \to q$ | $\to$I 2 |
| 4. | $(p \to q) \vee (q \to r)$ | $\vee$I 3 |
| 5. | $\neg q$ | Assumption |
| 6. | $q$ | Assumption |
| 7. | $\perp$ | $\neg$E 5, 6 |
| 8. | $r$ | $\perp$ 7 |
| 9. | $q \to r$ | $\to$I 6-8 |
| 10. | $(p \to q) \vee (q \to r)$ | $\vee$I 9 |
| 11. | $(p \to q) \vee (q \to r)$ | $\vee$E 2-4, 5-10, 1 |