

DUE: Tue 25 Sep

1. Equivalence Relation

Let R be a symmetric relation. Prove that R^* is an equivalence relation. Use relational calculus. Be as terse as possible.

2. Regular Expression Interpretation

Answer yes or no to each of the following questions, and give a brief argument justifying your answer.

- (a) Is $baa \in L(a^*b^*a^*b^*)$?
- (b) Is $b^*a^* \cap a^*b^* = a^* \cup b^*$?
- (c) Is $a^*b^* \cap c^*d^* = \emptyset$?
- (d) Is $abcd \in L((a(cd)^*b)^*)$?

3. Designing DFA

Construct DFA recognizing each of the following languages over the alphabet $\Sigma = \{a, b\}$

- (a) $L = \{w \mid \text{each } a \text{ in } w \text{ is immediately preceded and immediately followed by a } b\}$.
- (b) $L = \{w \mid w \text{ has } abab \text{ as a substring}\}$.
- (c) $L = \{w \mid w \text{ has neither } aa \text{ nor } bb \text{ as a substring}\}$.
- (d) $L = \{w \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s}\}$
- (e) $L = \{w \mid w \text{ has both } ab \text{ and } ba \text{ as substrings}\}$

4. Designing NFA

Construct NFA accepting each of the following. Use the power of the force (i.e., nondeterminism). Your automata should make guesses, and then check the input string. Be as lazy as possible while at the same time being correct. Do *not* attempt determinism for these without the help of a professional contortionist ...

- (a) $L = \{xy \mid x \text{ and } y \text{ are binary strings that each contain the same substring of length 3}\}$. (So, for example, 00101110100 is in L because of the repeat of 010, but 010000111 is not in L , since no substring of length 3 is repeated.)
- (b) $L = \{w \mid w \text{ is a binary string containing both the substring } 010 \text{ and the substring } 101\}$.
- (c) $L = \{0^i w 1^i \mid w \text{ is a binary string and } 0 \leq i \leq 4\}$.

5. Finite State Transducers

Your FSTs should be deterministic. (The example of a FST which was a "full adder" was discussed in class).

- (a) Text, Exercise 1.24
- (b) Text, Exercise 1.25

6. NFA \rightarrow DFA

Text, Exercise 1.16. You may use the algorithm given in lecture, or in the Text, Theorem 1.39

7. DFA \rightarrow RegExp

Text, Exercise 1.21. Use Kleene's Theorem as described in class

8. Closure Property

Text, Problem 1.40 (b)

9. Reversal Closure

Text, Problem 1.31

10. RegExp \rightarrow NFA

Text, Exercise 1.2.8.

11. Puzzle [Optional Extra Credit]

A man (M), wolf (W), goat (G) and cabbage (C) are on the left bank of a river. There is a boat large enough to carry the man and *only one* of the other three. The man wishes to ferry everyone to the right river bank, and the man can ferry each across, one at a time.

However, if the man leaves the wolf and goat unattended on either shore, the goat is history. Similarly the goat and cabbage cannot be left unattended by the man.

Design a DFA to find a way to to cross the river (using multiple crossings to and fro) without the goat or cabbage being eaten.

The problem is modeled by a finite state automaton. A *state* is a depiction of which of the participants is on the left bank, combined with which are on the right bank, separated by |. The *start* state is labeled " $MWGC|\emptyset$ ", since all participants are on the left bank, and there are no participants on the right bank. The only accepting state is labeled " $\emptyset|MWGC$ ". (There are 16 possible subsets of M, W, G, C that could be on one side of the river or other). States that are fatal to success and must be avoided are things like " $WG|MC$ " (goat gets eaten), " $GC|MW$ " (cabbage disappears), " $MC|WG$ ", etc.

The "input strings" to the automaton are the sequence of actions that the man takes. He may cross (in either direction) alone (input m), row across with the wolf (input w), row across with the goat (input g) or cabbage (c). For example, the input sequence gm results in the state sequence " $MWGC|\emptyset \rightarrow WC|MG \rightarrow MWC|G$ ". A g transition from the latter state is impossible (no goat on the bank with the man and boat). An m transition from the latter state results in $WC|MG$, which is valid but useless, since it simply returns to an earlier state.

- (a) Draw a transition diagram for this problem. You need not include "fatal" states, or undefined transitions.
- (b) Provide the string over $\{m, w, g, c\}$ that succeeds in crossing safely. Your sequence should be the *shortest possible* (minimal length) sequence of actions (There are two possible *minimal* solutions.)