CISC 462 Asn 2 solutions

1. a) No. M does not accept bbaa.

b) Yes. M accepts baaab.

c) No. Input is incorrect format. (Input should encode a DFA and a string.)

d) No. Input is incorrect format. (An input for A_{REX} should encode a regular expression and a string.)

e) No. $L(M) \neq \emptyset$

f) No. Input encodes only one DFA and is incorrect format.

g) Yes. $L(M) = L(M)$

$\boxed{2.}$ A decider for ALL_{NFA}:

Q = "On input $\langle M \rangle$ where M is a DFA

1. Convert M to an equivalent DFA M_D.
2. Let M_0 be a one state DFA recognizing Σ^*,
3. Run the decider F for EQ_{DFA} (Theorem 4.5) on input $\langle M_D, M_0 \rangle$.
   If F accepts accept, and else reject"

$\boxed{3.}$ a) i) f is not onto. There is no (integer) value x such that $f(x) = -1$.

ii) f is not one-to-one because $f(3) = f(-1) = 4$.

iii) No because it is neither onto nor one-to-one.

b) The following list contains all finite languages:

<u>Stage 1.</u> List all languages of cardinality at most one where the length of the longest string is at most one.

... ...

<u>Stage k.</u> List all languages of cardinality at most k where the length of the longest string is at most k. Omit languages that have been listed in stages 1, ..., k-1.

... ...

The enumeration contains all finite languages because any finite language L gets listed in stage m where m is the maximum of the cardinality of L and the length of the longest string in L.

4. Turing machine deciding A:

Q = "On input $\langle R \rangle$ where R is a regular expression:

1. Construct a DFA M_1 that is equivalent to R.

2. Construct a DFA M_2 that recognizes the complement of $\Sigma^* 0000 \Sigma^* + \Sigma^* 111 \Sigma^*$. This is possible due to closure properties of regular languages.

3. Construct a DFA M_3 such that $L(M_3) = L(M_1) \cap L(M_2)$. This is possible because regular languages are closed under intersection.

4. Decide whether $L(M_3) = \emptyset$ using Theorem 4.4. If yes, accept. If no, reject."

$\boxed{5.}$ We reduce A_{TM} to B.

Suppose TM P decides B. Construct following decider for A_{TM}:

$Q =$ "On input $\langle M, w \rangle$ where M is a TM and w a string

1. Construct TM M_w:

   $M_w =$ " On input string v

   1. Check whether $v = w$. If not, reject. Otherwise continue.

   2. Simulate M on v and answer what M answers"

2. Run decider P on input $\langle M_w, w \rangle$. If P accepts, accept. If P rejects, reject."

Why this works: $L(M_w) = \begin{cases} \{w\} & \text{if } M \text{ accepts } w. \\ \emptyset & \text{otherwise.} \end{cases}$

Thus M_w accepts some power of w if and only if M accepts w.

$\boxed{6.}$ $A_{US} = \{\langle M \rangle \mid M$ is a TM and M contains one or more useless states $\}$

We reduce A_{TM} to A_{US}:

Assume that TM N decides A_{US}. We construct a TM P that will decide A_{TM}.

Definition of P:

On input $\langle M, w \rangle$ (where M is a TM and w an input for it) P constructs a TM M' that operates as follows:

1. if the input for M' is $\langle M, w \rangle$ (= the fixed TM M and string w), simulate the computation of M on w
   else if the input is not $\langle M, w \rangle$, go to state q_{reject}
2. if the computation of M on w accepts, M' goes into a finite loop that goes through all non-rejecting states and ends in q_{accept}. (The loop can be made to work, for instance, by writing on the tape two consecutive symbols # (that are not used anywhere else), and the transitions on # are defined so that the tape head moves back and forth (left and right), and goes through all the states.)

After this P runs N on input $\langle M' \rangle$ and accepts if N rejects, and rejects if N accepts.

6. (continued)

Explanation why the construction works:

i) If M accepts w, then the computation of M' on <M, w> goes through all the states except qreject. The computation of M' on any other input uses qreject. Hence M' does not have useless states.

(ii) If M does not accept w, then M' does not accept any input and qaccept is a useless state of M'.

Hence M accepts w iff (if and only if) M' does not have useless states.