**Assignment 6 (100 points, Chapters 17, 20, 21, 28)**

**Submission: Please type your answers in this WORD file and submit to Tracs.** **Please also zip all the created \*.jff files into a single file and submit to Tracs.**

Note: You do not need to answer questions labeled with “self-study”, but you’re required to study them.

1. (10) The following is a high level description for a Turing machine *M*. List the first four elements of *L*(*M*) in lexicographic order (shortest first, then alphabetically if same length).

1. Move right.

2. Loop:

2.1. If the current symbol is x, change it to $ and move right. Otherwise exit loop.

2.2. Scan rightwards, past x’s and #’s to find y.

2.3. If y found, change it to # and move right. Otherwise reject.

2.4. Scan rightwards, past y’s and %’s to find z.

2.5. If z found, change it to % and move left. Otherwise reject.

2.6. Scan leftwards, past x’s, y’s, #’s and %’s to find $.

2.7. When finding the first $, move right, and go back to 2.1.

3. If the current symbol is #, more right. Otherwise reject.

4. Scan rightwards, past #’s and %’s, to find the blank symbol.

5. When finding the first blank symbol, move right and accept.

xyz, xxyyzz , xxxyyyzzz , xxxxyyyyzzzz

2. (20) Design a Turing machine *M* that decides the language *L* = {0*n*1*n* | *n* ≥ 1}

(1) Give a high level description of M in English.

On input string w

While there are unmarked 0s, do

Mark the left most 0

Scan right till the leftmost unmarked 1;

Mark the leftmost 1

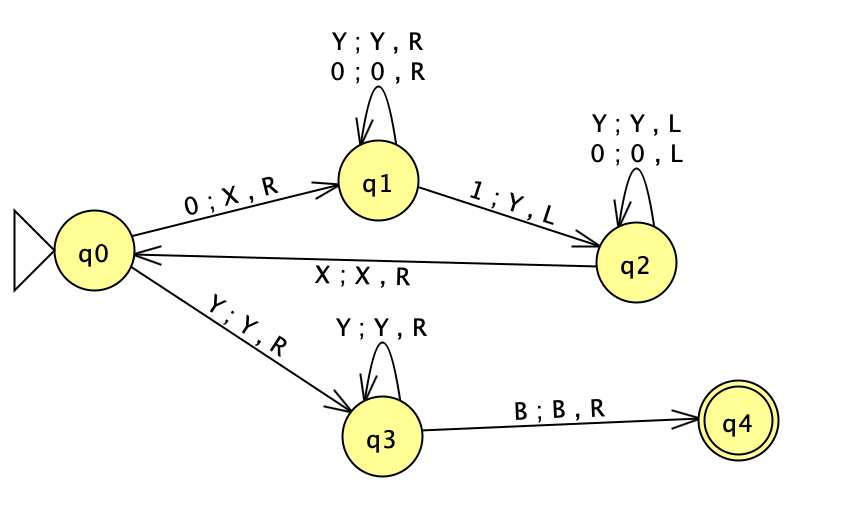
Done

Check to see there are no unmarked 1s;

If there are then crash

accept

(2) Define M with transition diagram. (create a6-2.jff and submit to TRACS, cut and paste figure below)



(3) In JFLAP, run your created Turing machine on the following list of testing strings (Click Input then Multiple Run. Input all the testing strings and click Run inputs). Indicate which of the testing strings are accepted. Your answer MUST be based on the actual running results from JFLAP.

Testing strings: ε,0,1,01,10,001,011,0011,0101,1010

Accept, reject, reject, accept, reject, accept, accept, accept, reject, reject

3. (10) Design a Turing machine *M* that decides the language *L* = {0*n*1*n* | *n* ≥ 0}.

(1) Give a high level description of M in English.

On input string w

While there are unmarked 0s, do

Mark the left most 0

Scan right till the leftmost unmarked 1;

If there is no such 1 then crash

Mark the leftmost 1

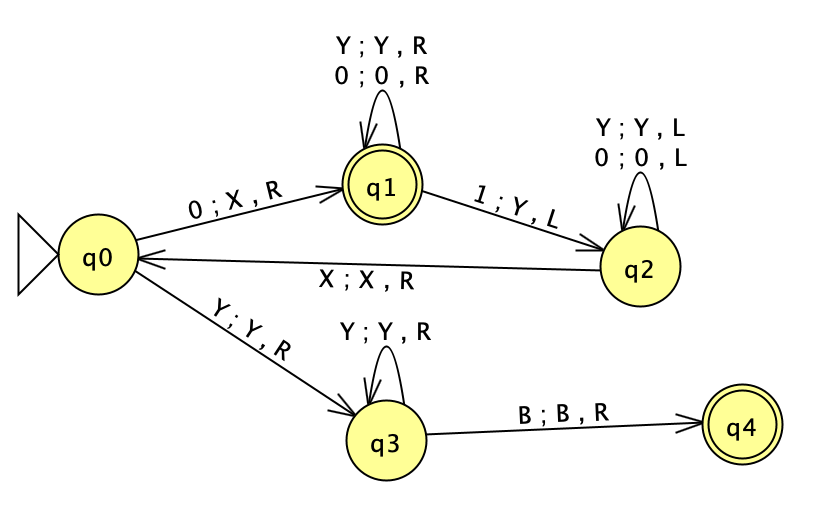
Done

Check to see there are no unmarked 1s;

If there are then crash

accept

(2) Define M with transition diagram. (create a6-3.jff and submit to TRACS, cut and paste figure below)



(3) In JFLAP, run your created Turing machine on the following list of testing strings (Click Input then Multiple Run. Input all the testing strings and click Run inputs). Indicate which of the testing strings are accepted. Your answer MUST be based on the actual running results from JFLAP..

Testing strings: ε,0,1,01,10,001,011,0011,0101,1010

Accept, accept, reject, accept, reject, accept, accept, accept, reject, reject

4. (30) True or False. Briefly explain.

(1). If L1 is not in D and L2 is regular, then it is possible that L1 ∩ L2 is regular.

True b/c regular language would be closed under intersection

(2). The union of two context-free languages must be in D.

True, bc closed under union

(3). If L1 ∩ L2 is in D then both L1 and L2 must be in D.

True, the intersection of language will be in D

(4). If L is in SD and its complement is context-free, then L must be in D.

False, context free not closed under complementation.

(5). If L is in SD then its complement must not be in D.

False – complement could or could not be in D, but will not be in SD

(6). If L1 is in D and L2 is in SD then L1 ∩ L2 must be in D.

False

(7). If L1 is in D and L2 is in SD then L1 ∩ L2 must be in SD.

True

(8). If L1 and L2 are in D, then L1 - L2 must be in D.

True

(9). If L1 and L2 are not in D, then L1 - L2 cannot be regular.

True if l1 is a subset l2 or l2 subset of l1 and l1 is not regular and l2 is not regular then l1-l2 is also not regular.

(10). If L1 and L2 are not in D, then L1 ∪ L2 cannot be in D.

True – languages l1 and l2, if they are context free, reg, decidable, they are closed under union. Therefore, if l1 and l2 are not in d, then l1 union l2 cannot be in d

(11). Every infinite language has a subset that is not in D.

True – every infinite set has an undecidable subset

(12). If ¬H were in D then every SD language would be in D.

True

(13). {<M> : L(M) is context free} is in D.

False

(14). {<M> : L(M) is not context free} is in D.

False

(15). If L1 is reducible to L2 and L2 ∈ D then L1 ∈ D.

False

(16). If L1 is reducible to L2 and L2 ∈ SD then L1 ∈ SD.

True

(17). If L1 is reducible to L2 and L1 ∉ D, then L2 ∉ D.

False

(18). If L1 is reducible to L2 and L1 ∉ SD, then L2 ∉ SD.

True

(19). If L1 is reducible to L2 and L2 ∉ D, then L1 ∉ D.

False

(20). If L1 is reducible to L2 and L2 ∉ SD, then L1 ∉ SD.

True

(21). (self-study) If L1 ∩ L2 is in SD then both L1 and L2 must be in SD.

False.

Let L1 be ¬H and let L2 be {a}. Then L1 ∩ L2 = ∅, which is semidecidable. But L1 is not.

(22). (self-study) If L1 and L3 are in D and L1 ⊆ L2 ⊆ L3, then L2 must be in D.

False

Let L1 be ∅ and let L3 be Σ\*. Both of them are in D. Suppose L2 is H, which is not in D.

Another example:

Let L1 = ∅. Let L3 = {<M>}. Let L2 = {<M> : M accepts ε}, which is not decidable.

(23). (self-study) Every infinite language has a subset that is not in SD.

True.

Let L be any infinite language. It has an uncountable number of subsets. There are only countably infinitely many semi-decidable languages (since there are only countably infinitely many Turing machines).

(24). (self-study){<M> : |L(M)| > 5} is in D.

False.

Rice’s theorem. The property in question is a nontrivial property of the SD languages.

(25). (self-study) {<M> : L(M) is in SD} is in D.

True.

The definition of an SD language is that it is accepted by some Turing machine.

Note that Rice’s theorem cannot be applied because this is a trivial property of SD languages (it’s like saying every SD language is in SD).

(26). (self-study) Rice’s Theorem tells us that {<M1, M2> : L(M1) ⊆ L(M2)} is not in D.

False.

This language is indeed not in D. But Rice’s Theorem doesn’t apply because the property in question is of an ordered pair of SD languages, not a single language.

(27). (self-study) Rice’s Theorem tells us that {<M> : M accepts all even length strings} is not in SD.

False.

This language is indeed not in SD. But Rice’s Theorem only tells us that it is not in D.

5. (15) Let R be the reduction from 3-SAT to INDEPENDENT-SET as discussed in class.

(1) Show the 3-CNF formula for which R builds the following graph.

*¬P P T*

*Q T Q S ¬Q S*

(-P v Q v T) ^ (-P v Q v S) ^ ( -Q v S v T)

(2) Show the graph that *R* builds for 3-CNF formula (A ∨ B ∨ C) ∧ (B ∨ C ∨ D). (Note: you may cut and paste the above graph and modify it.)

6. (15) Let R be the reduction from 3-SAT to VERTEX-COVER as discussed in class.

(1) Show the 3-CNF formula for which R builds the following graph.

*P ¬P Q ¬Q S ¬S T ¬T*

*¬P ¬P T*

*Q T Q S ¬Q S*

(-p V q V t) ^ (p V -p))^((q V -q)^(-p V q V s)^(s V -s))^((tV-t)^(t V q V s))

2) Show the graph that *R* builds for 3-CNF formula (A ∨ B ∨ C) ∧ (¬B ∨ ¬C ∨ ¬D). (Note: you may cut and paste the above graph and modify it.

7. (self-study) Show CLIQUE is NP-complete by first showing it is in NP and then showing it is NP-hard:

CLIQUE = {<G, k> : G is an undirected graph with vertices V and edges E, k is an integer, 1 ≤ k ≤ |V|, and G contains a k-clique}

